Terms and Conditions of Use of Digitised Theses from Trinity College Library Dublin

Copyright statement

All material supplied by Trinity College Library is protected by copyright (under the Copyright and Related Rights Act, 2000 as amended) and other relevant Intellectual Property Rights. By accessing and using a Digitised Thesis from Trinity College Library you acknowledge that all Intellectual Property Rights in any Works supplied are the sole and exclusive property of the copyright and/or other IPR holder. Specific copyright holders may not be explicitly identified. Use of materials from other sources within a thesis should not be construed as a claim over them.

A non-exclusive, non-transferable licence is hereby granted to those using or reproducing, in whole or in part, the material for valid purposes, providing the copyright owners are acknowledged using the normal conventions. Where specific permission to use material is required, this is identified and such permission must be sought from the copyright holder or agency cited.

Liability statement

By using a Digitised Thesis, I accept that Trinity College Dublin bears no legal responsibility for the accuracy, legality or comprehensiveness of materials contained within the thesis, and that Trinity College Dublin accepts no liability for indirect, consequential, or incidental, damages or losses arising from use of the thesis for whatever reason. Information located in a thesis may be subject to specific use constraints, details of which may not be explicitly described. It is the responsibility of potential and actual users to be aware of such constraints and to abide by them. By making use of material from a digitised thesis, you accept these copyright and disclaimer provisions. Where it is brought to the attention of Trinity College Library that there may be a breach of copyright or other restraint, it is the policy to withdraw or take down access to a thesis while the issue is being resolved.

Access Agreement

By using a Digitised Thesis from Trinity College Library you are bound by the following Terms & Conditions. Please read them carefully.

I have read and I understand the following statement: All material supplied via a Digitised Thesis from Trinity College Library is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of a thesis is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form providing the copyright owners are acknowledged using the normal conventions. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone. This copy has been supplied on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

Temporally Consistent Region Based Video Segmentation

A thesis submitted to University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Computer Science and Statistics,

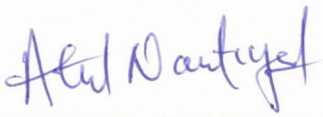
University of Dublin, Trinity College

August 2014

Atul Nautiyal

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

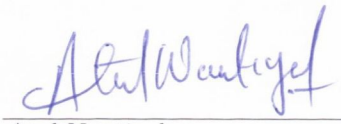


Atul Nautiyal

August 11, 2014

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.



Atul Nautiyal

August 11, 2014

Dedicated to

my beloved parents and Karanprayag

Acknowledgments

This thesis would not have been possible without the guidance and support of my supervisor Kenneth Dawson-Howe. I would sincerely like to thank him for his guidance, continuous encouragement, constructive criticism, generous help, endless patience and enormous freedom he gave me during the PhD.

My colleagues in *GV2* over past 4 years played a vital role in this work. I am grateful to them all for their support, advice, proof reading and general good company. A special thanks goes to Claudia Arellano, Zbigniew Edm Zdziarski, Donghoon Kim and Sayandeep Purkayasth for consistent moral support throughout and excellent advice.

I would like to express my deepest appreciation to Bachan Singh who gave me immense support since the day I stepped in Dublin to pursue my education. I would also thank my friends for providing support and friendship that I needed. I thank Arnab, Mithileash, Amit and Soumya for making my stay in Dublin wonderful and their belief in me. I especially thank Arunima Gulati for her support, patience and constant encouragement during my PhD. Last but not the least; I thank my parents Bhuwan Nautiyal and Premlata Nautiyal, my sister Archana and my brother Lava for their support and unconditional love.

Atul Nautiyal

University of Dublin, Trinity College

August 2014

Abstract

This thesis addresses the problem of segmenting a video sequence in a temporally consistent fashion, so that the labels assigned to particular region remain the same throughout the sequence.

Energy minimisation, and in particular α-expansion based label propagation, is a popular approach for addressing video segmentation. In this approach, labels of most similar object classes are assigned to pixels such that the energy of final libelling is minimised. Two major limitations of this approach are not being able to handle *non class* based segmentation and appearance of new objects in the current frame. In this thesis, we developed a novel method for label propagation in the forward direction using the α-expansion method. Our *spatio-temporal displacement constraint* enables the α-expansion to handle both *non class* and *physical class* based segmentation with lesser pixel to label comparisons. Appearance of new objects is handled by comparing colour properties of pixels and the existing objects during the label propagation. We extend our approach to learn the *physical explanations* (like occlusion or deocclusion) behind any changes in the segmentation results over time. Learned physical explanations also help us to determine occlusion boundaries.

To evaluate video segmentation we must assess both the accuracy of labelling per frame (spatial accuracy) and the consistency of labelling over the period of time (temporal consistency). We have developed three metrics; two for spatial accuracy and one for measuring the temporal consistency of the segmentation results. Our spatial accuracy metrics measure the accuracy by comparing segmentation results with the human annotated ground truth. Our temporal consistency metric does not require ground truth but successfully measures the consistency of the segmentation results over a block of frames.

Contents

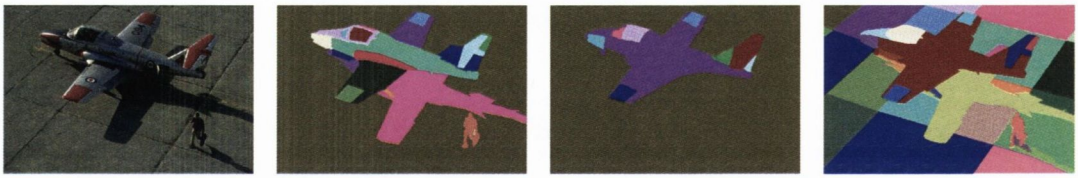
List of Tables

List of Figures

Chapter 1

Introduction

In computer vision, segmentation is the process of dividing a scene (image or video frame) into different meaningful regions which can further be used for scene interpretation. Segmentation is an ill posed problem because segmentation decisions are influenced by the depth of knowledge that any segmentation method has about the scene [3]. As a result, it is possible that different segmentation methods or human observers produce different but accurate segmentations of the same scene (Figure 1.1).



(a) Image (b) Annotator id: 1123 (c) Annotator id: 1130 (d) Annotator id: 1132

Figure 1.1: Segmentation is an ill posed problem. Three human annotators create different segmentations for the same image. Image and segmentation results are taken from the Berkeley Database [4].

Temporally consistent segmentation of video sequences is the basis for a number of applications in different domains, such as video tracking, video compression, video content based indexing, surveillance, artistic stylisation and video forensics. These applications rely on both the accuracy of segmentation in individual frames and the consistency of segmentation over time. Video segmentation results are considered to be accurate if regions do not contain boundaries of objects within them, and temporally consistent if the labels assigned to the pixels belonging to an object remain the same throughout the video. Due to factors like motion, noise, illumination, occlusion or deocclusion, segmentation methods can assign dif-

ferent labels to the same object or some parts of it. This makes it difficult for higher level applications to correctly interpret the scenes in a given video.

In video segmentation, accuracy and temporal consistency are two important requirements. To date, accurate segmentation of individual frames has received more attention than temporally consistent segmentation of video. A common belief is that accurate spatial segmentation of image regions will automatically enforce temporal consistency of regions in video frames. However, temporal consistency information can provide cues for video object detection and can also be used to improve video segmentation accuracy. The main objective of this research is to improve the state of the art in temporally consistent region based video segmentation.

There are two types of temporal consistency: short term and long term. Video segmentation results are short term temporally consistent if they are temporally consistent only for a small number of frames [5, 6, 7, 8, 9]. For short term temporal consistency, labels assigned to pixels belonging to the objects change quite often over the whole video sequence. Most of the block based video segmentation approaches deliver short term temporal consistency, i.e. over frames of the same block. For long term temporal consistency, segmentation results of the objects should remain temporally consistent while any part of the object is visible in the video sequence. There are two popular approaches that try to achieve long term temporal consistency: (i) tracking based [10, 11] and (ii) label propagation based methods [12, 13, 14, 15, 7]. Tracking based methods compute temporal information in advance. Segmentation is then performed by clustering spatially and temporally coherent pixels together. Label propagation based methods formulate image segmentation as an energy optimisation problem where segmentation is achieved by minimising the defined energy functions. In general, these methods produce better temporally consistent segmentation results in comparison with the segmentation methods based on two other approaches as detailed in chapter 2. Therefore, in our research we have developed an approach using label propagation methodologies to achieve long term temporally consistent segmentation results.

The result of label propagation is called the *label mask* or *segmentation mask*. Normally, a single label is assigned to pixels that are homogeneous according to some defined criteria, e.g. colour, texture and motion. A group of such pixels is called a *segment*. A *segment* should represent a complete physical object or a part of a physical object, but it should not have pixels belonging to more than one physical object.

1.1 Contributions

The novel contributions from this research work are as follows.

1.1.1 Spatio-temporal displacement constraint

Label propagation is a two step process. In the first step, a set of *learned* labels is produced by learning the properties, e.g., colour, texture, etc., of objects present in video using training on either test videos or ground truth data of a few (1 or 2) previous frames. Normally, each label of the *learned* label set represents a class of physical objects, for example, sky, car, grass, road, boat and sheep. We call this type of label propagation *physical class based* label propagation. In the *physical class based* label propagation methods, each label class has *unique* properties and represents a class of physical objects. Therefore, *segments* of a video sequence represent physical objects and in any frame, multiple instances (e.g., presence of many cows) of the same object class gets the same label. If different labels are used to represent multiple instances of a physical class then it can be referred to as *non class based* label propagation. In the second step, the best labelling is achieved by minimising a defined energy function. Energy minimisation methods often select local minimum due to inefficiency of computing the global minimum. In general, local minimisation techniques are sensitive to the initial estimates and produce a local minimum far from the global minimum. α-expansion is a powerful graph cut based algorithm for energy minimisation [16]. α-expansion does not depend on initialisation and guarantees local minimum within two factors of the global minimum. After starting from an initial labelling the algorithm changes the labelling iteratively. A change which leads towards the minimum local energy is selected at each iteration. Final labelling is said to be achieved when further minimisation of energy is not possible. In this research, we use α-expansion for minimising our energy function (Chapter 3).

α-expansion based label propagation methods propagate labels by comparing *each* pixel of the current frame with *each* label of the *learned* label set. Hence, any label of the *learned* label set can be assigned to any pixel of the current frame. In general, in a video sequence, any object present in the current frame can only move by a small distance in the next frame. Therefore, label propagation methods should not compare each pixel of the current frame with each label of the *learned* label set. They should only compare each pixel with the labels of objects which were present in the pixel's spatial neighbourhood in the previous frame. We

call this constraint *spatio-temporal displacement constraint* and propose a novel method that enables label propagation methods to produce robust results with lesser number of comparisons (pixel to label). The spatio-temporal displacement constraint is a novel contribution which allows α-expansion [16] to:

- handle *non class based* segmentation, and
- produce results with lesser number of comparisons (pixel to label).

Spatio-temporal displacement constraint is one of the major contributions of this thesis and is discussed in detail in section 3.3.

1.1.2 Handling new objects

Label propagation uses the *learned* label set for assigning labels to pixels of the current frame I_t. Hence, in the case of appearance of new (not present in the *learned* label set) objects in I_t, label propagation will assign incorrect labels to the pixels belonging to the new objects [17, 18].

We have developed a novel pre-processing step to deal with new objects. Our approach is a four step approach. In the first step, a new label *void* is added in the *learned* label set to represent new objects. In the second step, an energy function (unary potential) is computed such that during the optimisation it favours assigning *void* label to a pixel if the colour of pixel is not *statistically* similar to the colour of atleast one label present in the *learned* label set. In this research, we have limited our determination of new objects to those which have different colour distributions from the existing objects. In the third step, pixels marked as new objects are clustered in *Colour* + XY space. In order to avoid segmentation errors, only clusters bigger than a *minimum area* are considered as new objects. In the last step, our algorithm learns properties of these clusters and re-runs the label propagation method for the current frame with the new *learned* set of labels. Section 3.6 describes this contribution comprehensively.

1.1.3 Improving label propagation results for rigid objects

Our label propagation method uses properties, i.e., colour and texture of labels present in the *learned* set for assigning labels to each pixel of the current frame. If the properties of pixels

belonging to the two neighbouring *segments* are similar, our label propagation may misclassify some or all of the pixels from these *segments*.

The segmentation results achieved using the label propagation method may be incorrect but can be used to learn new cues such as motion and shape of the *segments*. These new cues can further be used to improve the label propagation results. We have developed a novel method which improves the label propagation results for rigid objects using shape and motion cues learned with the help of the segmentation results of the current and previous frames. This contribution is covered in chapter 4.

1.1.4 Learning physical explanations behind changes in segments shapes and occlusion boundaries

Rigid objects present in a video may exhibit motion, such as translation, rotation and scaling. These moving objects can occlude or deocclude themselves and other objects. Occlusion is an event where, in the view of an observer, objects disappear (completely or partially) behind other objects. Deocculsion is an event where new parts of the objects appear from behind the other objects. Both occlusion and deocclusion changes the visible shape of objects in the view of an observer. We have developed a method for categorising changes in the shape of *segments* as occlusion or deocclusion. We refer to it as *learning the physical explanation*. Along with the physical explanation, we have developed a novel algorithm for detecting occlusion boundaries present in any frame. Occlusion boundaries are boundaries present between the foreground and the background *segments*. This contribution is detailed in chapter 5.

1.1.5 Evaluation metrics

We have developed three metrics: spatial consistency, size consistency and temporal consistency metrics to measure the accuracy and temporal consistency of the segmentation results. In general, human annotated ground truth data is used to measure the accuracy of segmentation results. In such data, often segmented regions or *segments* reflect the human perception of the objects (figure 1.1). This level of division is tremendously difficult to achieve for any segmentation algorithm, as it requires object level knowledge or understanding. Therefore, a single region (object) present in the ground truth data is typically segmented as several regions by the segmentation algorithm (over-segmentation). The evaluation process should not penalise such over-segmentation, however, it must consider the number and size of over-

segmented regions to measure the segmentation results accurately. Considering these requirements, we have developed two metrics: spatial consistency and size consistency for measuring the accuracy of the segmentation results.

Another metric, the temporal consistency metric is developed for measuring the temporal consistency of segmentation results by comparing label consistency of the pixels in both forward and backward directions. Existing evaluation metrics are detailed in Section 2.4.

Spatial consistency metric

The spatial consistency metric measures the accuracy of segmentation results against the ground truth. Normally, human annotated ground truth is a high level (object based) segmentation and consequently, over-segmentation is expected with segmentation methods. Thus, there is a need for over-segmented regions to be clustered together before comparison with ground truth objects. Our metric developed for measuring the spatial consistency is detailed in section 6.1.

Size consistency metric

The spatial consistency metric does not consider the size and number of over-segmented *segments*. Let us assume that a segmentation algorithm divides a ground truth object into *segments* of size 1 pixel each. Our spatial consistency metric will cluster the *segments* together without considering their sizes. Hence, this segmentation will get maximum score for the spatial consistency. Therefore, the spatial consistency metric alone is not enough to measure the segmentation quality. We therefore introduce another metric called *size consistency* to penalise the segmentation quality based on the number and the sizes of regions. This metric is detailed in section 6.2.

Temporal consistency metric

From a review of existing literature, it is apparent that there is a lack of robust methods and metrics for measuring the temporal consistency of video segmentation. Temporal consistency of video segmentation results can be measured using ground truth with pixel to pixel correspondence information. Such correspondence information enables evaluation metrics to check the consistency of labels assigned to pixels over time. However, creating such detailed ground truth is a tremendously difficult task. We have developed a novel method to measure the

temporal consistency of the segmentation results. Our method compares the segmentation results of the current frame only with the past and future segmentation results without need of the ground truth data. The developed algorithm is discussed in section 6.3. Temporal consistency metric is another of the major contributions of this thesis.

1.2 Report outline

In chapter 2, we present a review of the state of the art for three types of region based video segmentation methods (block based, tracking based and energy minimisation based label propagation methods) and for evaluation metrics. We present the basics of the α-expansion algorithm along with the algorithms developed for label propagation in the forward direction and those for handling new objects in chapter 3. Our algorithms for improving the label propagation for rigid objects is presented in chapter 4. In chapter 5, we present our algorithms for learning physical explanations and occlusion boundaries. In chapter 6, we present three new metrics for measuring the accuracy and the temporal consistency of segmentation results. We present our results in chapter 7. A summary of this research work and directions for future work are presented in chapter 8.

In the domain of video making and analysis, a shot refers to a sequence of frames with continuous action. A shot change refers to a sudden change of frame content. The proposed methods will work only on video sequences consisting of frames from a single shot where videos can be captured with a moving or static camera. Label propagation presented in chapter 3 uses brightness, colour and texture cues. However, handling appearance of new objects uses only colour and brightness cues. Our algorithm learns new cues like shape and motion after the initial segmentation and uses these new cues along with colour and brightness to improve the segmentation results in chapters 4 and 5. Techniques detailed in chapters 3 and 6 are applicable to non rigid objects as well as rigid objects. However, techniques detailed in chapters 4 and 5 do not deal with non rigid objects.

Chapter 2

State of the art

This chapter presents state of the art of the region based temporal consistent video segmentation and evaluation metrics (Section 2.4). In our review of region based temporal consistent video segmentation, we focused on three types of video segmentation methods; *block based* (Section 2.1), *tracking based* (Section 2.2) and *energy minimization based label propagation methods* (Section 2.3). Video is a collection of image frames. Therefore, the simplest way of segmenting any video is to segment each frame individually. However, this technique lacks temporal coherence because each frame gets segmented independently. A possible solution is to segment the image frames into groups. *Block based* video segmentation methods try to achieve both spatial and temporal coherence by segmenting more than one frame simultaneously. These methods first divide the video in small blocks of temporal and spatially consistent frames and then segment frames of each block. Block based video segmentation methods are unable to produce temporally consistent results over long spans of the time (Section 2.1.4).

Tracking based video segmentation methods attempt to improve temporal consistency of segmentation by computing temporal information in advance. In the first step, these methods compute the motion trajectories of feature points or pixels over the whole video sequence. In the next step, segmentation is achieved by clustering the pixels around the trajectories of similar feature points. Temporal consistency of these methods is discussed in section 2.2.1.

Label propagation based methods learn the information, e.g., colour distributions, shape and texture about objects present in video sequence by training either on the training video sets or on the previous video frames. Later this information is used to assign a label to each pixel of the current frame using the energy minimization framework. We have observed that despite the current limitations (section 2.3.3), these methods especially the α-expansion based

methods produce more accurate and more temporally consistent results than the block based video segmentation methods. These methods also have more general purpose applicability in comparison to the tracking based approaches.

2.1 Block based video segmentation methods

Video segmentation methods rely on spatial and temporal coherence of objects present in video to produce more accurate and robust segmentation results. Spatial segmentation of a frame is same as image segmentation. Therefore, video segmentation methods inherit algorithms from the image segmentation domain for spatial segmentation of frames. The objects in a video sequence experience motion, occlusion and changes in illumination. Therefore, it is very hard to enforce spatial and temporal coherence of objects over a long period of time. To overcome these limits, volumetric techniques try to achieve a short term spatio temporal coherence by decomposing the video sequence into smaller blocks of frames. In the simplest case, each block can hold a single frame [19]. However, lack of temporal information in the one frame approach may cause unstable segmentation results. A two frame per block approach tries to address this issue by segmenting two frames simultaneously [20, 21, 5, 8]. On the negative side, this model only uses one directional (either backward or forward) temporal information for segmentation of frames. A common approach is allocating three frames per block where central frame works as a reference frame [6, 7]. The three frame approach uses motion and localization cues from both the directions (backward and forward) to produce segmentation results.

A few initial segmentation approaches do not follow the above described fixed number of frames per block allocation mechanism. For example, Huang et al. [22] used two frame approach to compute optical flows for patches and then used three frames to create initial spatial-temporal segmentation. In another example, after generating the particles (feature points) trajectories using whole video sequence, Silva and Scharcanski [10] used a 2-frame approach for spatial clustering of the trajectories.

2.1.1 Segmentation of blocks of frames

Video segmentation is a process of labelling pixels of frames based on cues like colour, texture, motion, shape and higher order co-occurrences. Generally cues used in the image and video

segmentation process can be divided into three levels: low, medium and high level cues. Low level cues, such as colour and brightness consider pixels directly; medium level cues, such as region continuity and shape consider the properties of whole regions and; high level cues like co-occurrence relationships, for example, boat-water and airplane-sky consider scene properties for labelling the pixels [23].

After dividing video in blocks of frames (Section 2.1), video segmentation methods perform segmentation at the block of frames. Normally, each block of frames is segmented in two steps; initial segmentation and final segmentation. The initial segmentation divides the block of frames into small spatio temporal homogeneous regions using low level cues like, colour and brightness, and then the final segmentation methods merge/split these initial segments using motion coherence. However, there are some video segmentation methods which do not use the initial segmentation step and directly go for the final segmentation of blocks.

The motivation behind the initial segmentation is to create small local, coherent and boundary preserving groups of pixels to enhance the overall segmentation quality and to reduce the required computation power for the subsequent video processing. Initial segmentation divides the frames of a video block into small homogeneous spatio-temporal over-segmented regions based on a set of features while preserving objects boundary information. Normally, this initial segmentation is obtained by generalising the image segmentation methods into the $3D$ (spatial + time) domain.

Initial segmentation of a video block produces result as a $3D$ (spatial + time) graph. Initial segmentation should preserve the region boundaries but it is quite possible to lose region boundaries in this process. Therefore, another option is to skip the initial segmentation step and then use pixels directly as nodes of the graph for final segmentation. For example, Shi and Malik [20] initialised a weighted spatio temporal graph using pixels of two consecutive frames as nodes. In another work, Zhang and Ji [7] used pixels to construct a 3D conditional random field (CRF) model as they suggest the initial segmentation techniques may fail to find accurate objects boundaries. However, considering pixels as initial segments for further segmentation, demands enormous processing power. The goal in the final segmentation is to produce more meaningful spatio-temporal coherent regions by partitioning this input $3D$ graph.

Block based methods extensively rely on the neighbourhood of pixel information for categorising it into both spatial and temporal directions. Graphs provide a wonderful framework

to represent this neighbourhood information. Therefore, these methods use graphs to represent the input frames. We can divide the block based method in two categories; pair wise graph (Section 2.1.2) and hypergraph (Section 2.1.3) based methods.

2.1.2 Pairwise graph based methods

Pairwise graphs are very popular in the image and video segmentation because of their simplicity. These graphs are called pairwise because they capture only the pairwise relationship between its nodes. Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a undirected graph where vertices $\mathrm{v}_i \in \mathcal{V}$ represents the set of elements (pixels or superpixels) to be segmented and $(\mathrm{v}_i, \mathrm{v}_j) \in \mathcal{E}$ represents the set of edges between neighbouring vertices. A non negative weight $\mathrm{w}_{\{i,j\}}$, which captures the dissimilarity between neighbouring vertex pair v_i and v_j, is assigned to each edge $(\mathrm{v}_i, \mathrm{v}_j) \in \mathcal{E}$. A segmentation process divides this graph into different clusters of vertices such that elements within a cluster should have similar feature values and elements in different clusters should have dissimilar feature values.

Generalised Felzenszwalb and Huttenlocher's [24] image segmentation method provides a simple and effective graph based $3D$ (XY + time) segmentation framework [5, 9]. Felzenszwalb and Huttenlocher used a simple, intensity based predicate to merge graph nodes [24] and proposed feature points to deal with occlusion. Generally, graph based clustering methods provide no mechanism to control the cluster size. To address this behaviour, Galmer and Huet [5] used a size dependent threshold with the graph based methods for segmenting the first frame. This threshold regulates region size by allowing fewer merges as region size increases.

With the graph representation, segmentation problem can be treated as graph partition problem. Wu and Leahy [25] used graph cuts for the image segmentation. A cut partitions the nodes $\mathcal{V}$ of graph $\mathcal{G}$ into two disjoint subsets. Their method try to minimise the sum of edge weights across cut boundaries. However, this graph cut method was biased towards small regions. Shi and Malik [26] addressed this biased behaviour of graph cut by taking a normalised weight across the cut. Their method *Normalised Cut* (*NCut*) is the most popular graph cut method. Many other segmentation methods use *NCut* as a basis. Popular graph cut based methods use eigen vectors of an affinity (edge weights) matrix to partition the graph. In *NCut*, Shi and Malik [26] used eigen vectors with first k smallest eigenvalues to partition the $2D$ graph into k partitions, by finding the splitting points such that the *NCut* is minimised. Each partition is further repartitioned if *NCut* is less than a pre-specified stability

value. Later Shi and Malik generalised *NCut* for video domain [20].

The Superpixel algorithm of Ren and Malik [27] is a popular *NCut* based segmentation algorithm to produce a large number of small, compact quasi-uniform segmented image regions. In their method, edge affinity is calculated by connecting pixel's contour and texture cues. Levinshtein et al. [28] pointed out most of the initial segmentation algorithms, like local variations of graph-based algorithms [24], have no control over the numbers of *segments* and also produce *segments* of irregular shape and sizes. To overcome these limitations they proposed *TurboPixels* algorithm that produces lattice-like structure of compact regions by dilating seed pixels to adapt the local structure with roughly linear time complexity. Using geometric flow based algorithms, *TurboPixels* assures uniform sized, compact, connective, edge preserving (turbopixels don't have any object boundaries inside them) and non overlapping *segments*. However, *Turbopixel's* control over number of *segments* comes with a new limitation, as the number of required initial *segments* must be known in advance.

Multiscale segmentation is another approach to increase the accuracy and the speed of the segmentation methods [29, 30, 31]. In the multiscale approach, segmentation at lower resolution tries to capture the global information and segmentation at higher resolution captures the local information very accurately. Furthermore, segmentation results of the lower resolutions are propagated hierarchically to the higher levels to produce the final results. For example, Cour et al. [30] proposed a linear time, multiscale graph decomposition with normalised cuts for image segmentation. This multiscale graph decomposition method is adopted in video segmentation for initial segmentation because of its accuracy and linear time complexity [22, 31].

Instead of graph cuts some methods use simple merging predicates to produce the final segmentation [5, 9]. In these methods, the merging predicate and the order of merging are two important parameters. The merging predicate is a statistical test to decide the merging/splitting of the graph nodes. On the other hand, the merging order decides the order of merging for the nodes [32]. McDiarmid's inequality has been commonly used for merging predicate calculation [32, 8, 33]. Galmar and Huet [5] used a global threshold with a local merging predicate to control the size of spatio temporal *segments*. For simplicity, each edge weight is compared against the internal mean weights of the participating components rather than their maximum weights [24]. This method also categorises *segments* as static or moving. However, all parameters of this segmentation methods including the global threshold, are manually selected. In their work, Hassani et al. [8] used a intensity/colour based statistical

test to partition a frame into initial patches.

Stein and Hebert [2] proposed a watershed based over-segmentation, driven by the output of a learned, statistical and multi-cue edge detector after non-local maximum suppression. An edge map of the reference frame is constructed using the *Pb* edge detector [34]. The *Pb* edge detector is selected because it considers colour, brightness and texture cues simultaneously. The watershed is preferred over methods relying on normalised cuts for its simplicity, speed and more regularly shaped *segments*.

2.1.3 Hypergraph based methods

Graph based methods normally compute edge affinity by combining many weighted features. However, edge affinity calculated as a weighted sum of features may lead to loss of some vital information. Moreover, when affinity relations are not pair wise but rather of higher orders, simple graphs are unable to create good clusters. In clustering, sometimes three or more data points should be analysed together to determine if they belong to the same cluster or not [35]. A special kind of graph, hypergraph, represents this higher order relationship of data.

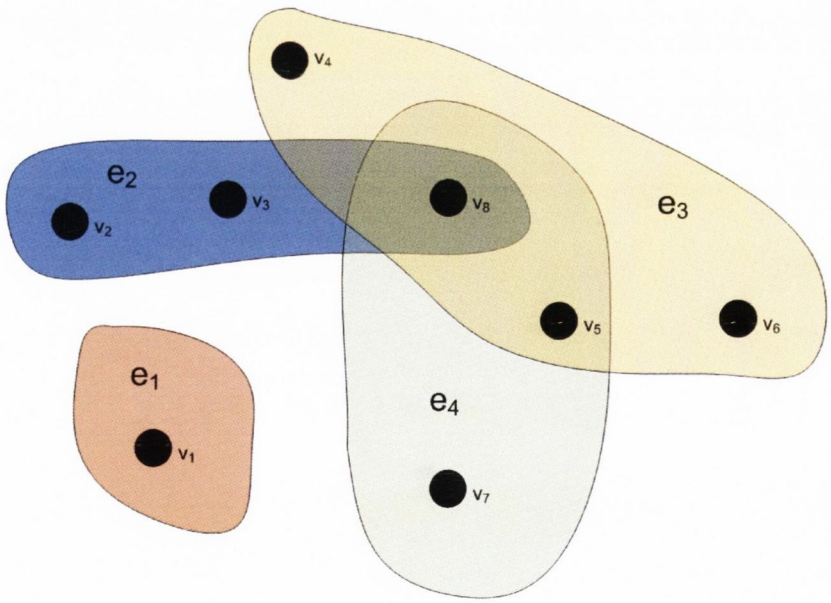


Figure 2.1: Example of a hypergraph with eight vertices $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and four hyper edges $\mathcal{E} = \{e_1, e_2, e_3, e_4\} = \{\{v_1\}, \{v_2, v_3, v_8\}, \{v_4, v_8, v_5, v_6\}, \{v_8, v_5, v_7\}\}$. Each hyper edge is highlighted in different colour.

A weighted undirected hypergraph $\mathcal{H}$ is a pair $(\mathcal{V}, h)$. Here $\mathcal{V}$ is the set of vertices of $\mathcal{H}$, and subsets of $\mathcal{V}$ are known as hyperedges $\mathcal{E}$. A hyperedge can connect more than two vertices, therefore a hyperedge is a subset of vertices. Degree of any vertex is sum of all contributing

hyperedges. The function h associates non-negative weights with each hyperedge. Figure 2.1, shows an example of hypergraph with eight vertices $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and four hyper edges $\mathcal{E} = \{e_1, e_2, e_3, e_4\} = \{\{v_1\}, \{v_2, v_3, v_8\}, \{v_4, v_8, v_5, v_6\}, \{v_8, v_5, v_7\}\}$. Agarwal et al. [35] were first to introduce hypergraphs in the computer vision for the image segmentation. However, to partition the hypergraph, their method approximates a weighted graph from the hypergraph. Later, Zhou et al. [36] extended spectral clustering methodologies (like *NCut*) of undirected graphs for the hypergraphs. Their normalised hypergaph cut is a popular algorithm for partitioning hypergraphs [22, 31].

Hypergraphs are gaining popularity in video segmentation in recent years because of their ability to capture higher order relationships. For example, Huang et al. [22] model the spatio temporal over-segmented image patches using hypergraphs. This method performs spectral analysis [36] on optical flows and motion profiles of patches to set the hyperedge weights. Furthermore, Huang et al. [22] emphasise hyperedges that contain moving objects by assigning them larger weights. Generally, hypergraph construction and clustering algorithms suffer from higher time complexity in comparison to pairwise graph based algorithms.

To deal with higher time complexity multilevel framework for hypergraphs has been proposed [37]. In the multilevel approach, initial partitioning has been computed over the coarsest hypergraph using eigen vectors. Afterwards, partitions of the coarser hypergraph are projected to the next finest level of the hypergraph.

2.1.4 Temporal consistency issues with block based video segmentation methods

We have identified following major issues when temporal consistency of block based methods break down:

Use of low level cues: In a video sequence, it is very likely that photometric properties and shape of objects will change over the time. Block based approaches use only information of pixels (low level cues) for the segmentation. However, this level of information is not sufficient for handling changes occurring at higher levels. As a result, these methods often mislabel pixels present near the object boundaries. Moreover, block based methods do not provide any mechanism to enforce the temporal consistency at object level.

Computational restrictions: Accuracy and temporal consistency of these methods depend on the number of frames used per block. Normally better segmentation results are achieved by increasing the number of frames per block. However, there are memory and processing power restrictions on the number of frames that a segmentation method can process simultaneously.

Region merge/split: Region merge is an event when two or more spatially disconnected regions/objects of similar properties get merged at some point of the time t and then stay together for considerable amount of time. Region split is an opposite event of region merge. In the region split, pixels of any region/object split into two or more motion coherent regions/objects. Lets assume at time t region s (label) splits into two regions s (label) and $s1$ (label). In the paradigm of temporal consistency, segmentation method must assign different labels to the pixels belonging to both the objects from start. This can be achieved only if temporal information over the whole video sequence is available in advance. Block based methods use limited temporal information for segmentation of the block of frames. Therefore, change happening due to the region merge/split outside the current block do not make any effect on the segmentation results.

2.2 Tracking based video segmentation methods

Lack of temporal information beyond the boundaries of a block is one of the main problems for the block based methods. A possible work around is to collect the temporal information present in all frame of the video in advance and then use it to for segmentation of each frame. Tracking based methods work on this principle.

Tracking is following any object of interest over time. In the video domain, tracking of an object can be made possible in two ways; tracking of all pixels belonging to an object or tracking only some specific features belonging to the object. To track the objects present in a scene, both approaches compute the motions of objects and establish a correspondence of every pixel or feature point respectively. Tracking methods which consider all pixels for tracking are called dense or direct methods [38]. Direct methods perform both motion computation of objects and establishing correspondence of every pixel simultaneously. In contrast, feature based methods used only a lesser number of points (in comparison to the number of pixels) per object for tracking. Therefore, these methods are often called sparse methods. Sparse or

feature based methods first extract a sparse set of features from each frame independently and then compute their motion by establishing correspondence for every feature point [39].

Segmentation methods based on direct tracking approaches methods use all pixels for tracking process. Computing motion of pixels especially in the presence of occlusion is most critical task for any tracking based approach. Optical flow is a popular choice in block based video based segmentation [6, 20, 9, 22, 2]. However, segmentation approaches based on direct methods generally avoid optical flow for motion computation because results of optical flow based methods are very ambiguous near object boundaries. Normally, tracking methods prefer motion models with higher degrees of freedom like affine motion models to model pixels motions [21, 10, 11]. In their work Kumar et al. [11], divided a frame into patches of 3×3 pixels and used consecutive frames to find parameters of affine models of every patch. In consecutive frames of a video sequence large motion is not expected. Therefore authors restrict the motion model to a small number of transformations. A joint probability distribution is solved to find out the motion parameters of each patch. A final segmentation is achieved by clustering neighbouring patches which are moving together. In general, direct tracking approaches produce broken (unlinked) trajectories for moving objects which remains static for a long time before moving again. An unlinked trajectory is a trajectory of pixels or features which is not linked with any other trajectory present outside its lifespan. To handle this situation Zhong and Chang [40] compare regions present in unlinked trajectory with the regions of other trajectories. An unlinked trajectory gets linked with a trajectory which shares the maximum number of common regions.

In contrast to the direct methods, feature based methods track a lesser number of points over frames. Feature based methods represent any object by its shape and appearance features like points, geometric shape, contours, colour, texture and probability density [41]. These methods choose features in each frame and then feature trajectories are created by solving feature correspondence. Temporal consistency of feature based segmentation methods depend highly on the quality of both feature points and feature trackers. In their work Marc et al. [21] used an affine motion model and a mesh motion model for tracking of pixels and feature points. The affine model learns motion of pixels and then a probability function is used to cluster pixels of similar motion in different classes. In any class, pixels with high probability are called seed pixels. A mesh model then tracks the seed pixels within a set of frames and produces seed trajectories. In the last step, a clustering function merge/split the previous

clusters into groups of coherent affine motion models and trajectories. This model only uses motion information for video segmentation so its performance may suffer in textured regions. Silva and Scharcanski [10] deal with this problem by considering the pixel neighbourhood's spatial and motion coherence as well. Their method uses particle tracking to create initial clusters. A hypergraph uses these initial clusters as its nodes and then produce final spatial and coherent clusters. Although this approach performs well in textured regions but the complexity of hypergraph will increase rapidly with the increase in number of frames used for particle tracking.

2.2.1 Temporal consistency issues with tracking based video segmentation methods

Tracking based segmentation methods have temporal information over the whole video sequence in advance. Therefore, these methods can deal with region merge/split issues. Low level cues like colour and brightness are used with temporal information of pixel/feature points for clustering similar pixels. As a result, tracking based methods are also affected by changes happening at higher levels (Section 2.1.4). The biggest problem with these methods is that they are only suitable for highly textured videos where each *segment* can have significant number of feature points in each frame. Performance of tracking based methods decline rapidly in non textured regions where the number of feature points is near to zero.

In section 2.3, we discuss label propagation based methods which use all; low, medium and higher level of cues gathered by training either on previous segmentation results or on training video set for labelling the pixels of video frames.

2.3 Label propagation based video segmentation methods

Video segmentation can be seen as a label propagation problem where labels existing in previous frames are propagated to label each pixel p of current frame I_t. A good labelling is achieved when final segments are spatially smooth and differences among segments and classes properties, e.g., colour, texture and shape, are minimum. The energy minimisation framework can provide a natural solution for this kind of optimisation problem and the most favourable labelling can be achieved by minimising the defined energy function.

Energy minimisation methods often select local minimum due to inefficiency of computing

the global minimum. In general, local minimisation techniques are sensitive to the initial estimates and choices of energy functions. As a result, a local minimum can be arbitrarily far from the optimum not conveying any of the global image properties [16]. Many optimisation techniques like Iterated Conditional Modes (ICM) [42] and simulated annealing [43] move towards the local/global minimum by changing label of a single pixel at a time. ICM is a greedy method and it convergences to a local minimum by assigning the label for which value of the selected energy function is minimum to each pixel. Simulated annealing approach searches the global optimum in the entire search space. Simulated annealing inspiration come from the process of slowly cooling molecules to form a perfect crystal. It requires exponential time for minimising an arbitrary function thus making it impractical for energy functions with several variables (labels).

The α-expansion and the $\alpha\beta$-swap are the two most popular move making algorithms for energy minimisation [16]. After starting from an initial labelling both algorithms change (move) the labelling iteratively. These algorithms can change label of more than one pixel in any move. A change which leads toward lesser energy is selected at each iteration. Final labelling is said to be achieved when further minimisation of energy is not possible. α-expansion does not depend on initialisation and guarantees local minimum within two factors of the global minimum in polynomial time. Therefore, we have used the α-expansion to minimise our energy function.

The general form of energy function $E(x)$ is composed of two energies, data energy (E_{data}) and smoothness energy (E_{smooth}) [16, 44, 45, 46, 47, 48, 49, 14, 12]:

$$E(x) = E_{data}(x) + E_{smooth}(x) \tag{2.1}$$

where, $E_{data}(x)$ measures the disagreement between labelling x and the observed data; and E_{smooth} measures the extent to which x is not piecewise smooth (spatially). Both $E_{data}(x)$ and $E_{smooth}(x)$ can be defined by the unary and pairwise terms of the *Gibbs* energy function. Hence, E_{data} can be written as the sum of data costs over all pixels:

$$E_{data}(x) = \sum_{p \in \mathcal{P}} \mathcal{D}_p(x_p) \tag{2.2}$$

where $\mathcal{D}_p(x_p)$ is a function defined on observed data that measures the cost of assigning label x_p to any pixel p and $\mathcal{P}$ represents all pixels of the frame.

In terms of the *Gibbs* energy, $E_{smooth}(x)$ can be written as the sum of pairwise potential functions:

$$E_{smooth}(x) = \sum_{\{p,q \in \mathcal{N}\}} \mathcal{D}_{p,q}(x_p, x_q) \tag{2.3}$$

where $\mathcal{N}$ defines the neighbourhood of pixels and $\mathcal{D}_{p,q}(x_p, x_q)$ measures the cost of assigning labels x_p and x_q to pixels p and q, i.e., $\{p, q\} \in \mathcal{N}$. $\mathcal{D}_{p,q}$ term encourages spatial smoothness by assigning the same label to both the pixels p and q. However, at borders where neighbouring pixels often represent different physical objects, $\mathcal{D}_{p,q}$ should avoid this behaviour. Such functionality can be achieved if $\mathcal{D}_{p,q}$ has the discontinuity preserving property. In the next sections, we present review of the two main methods used for energy minimisation which are loopy belief propagation and graph cut based methods.

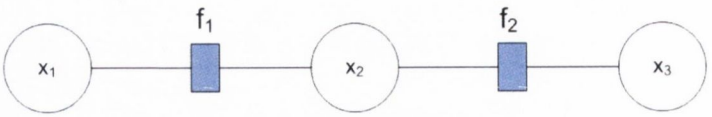
2.3.1 Loopy belief propagation

Loopy belief propagation (LBP) minimises the energy equation 2.1 by passing messages around the graph defined by the image pixels. Image pixels represent the graph nodes and the edge connecting any two nodes represents the compatibility between them. At each iteration, each node sends and receives messages from its neighbours. The length of each message can be given by the number of possible labels. Belief propagation methods compute belief at each node using neighbours messages and the local data (e.g, intensity, colour etc.). Finally, the labelling that minimises the energy function $E(x)$ at each node is selected. There are two main variants of the loopy belief propagation methods max-product based [50, 51, 52] and sum-product based [53]. The max-product based loopy belief propagation method tries to find out the labelling x that maximises the global energy function. The other variant, sum-product based loopy belief propagation method computes the negative logarithm of the marginal probability of each node to find out the optimal labelling x.

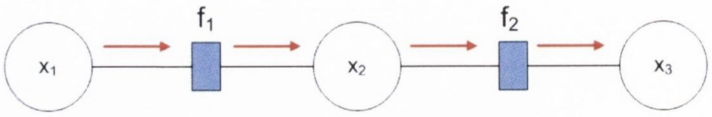
Loopy belief propagation methods compute local marginal probabilities using factor graphs. A factor graph $\mathcal{G} = \{X, F, E\}$ consists of variable vertices X, factor vertices F and edges E. Factor graph helps in factorisation of a function [54]. Factor graph shown in figure 2.2a factorises a function $p(x)$ as

$$p(x) = \frac{1}{Z} f_1(x_1, x_2).f_2(x_2, x_3) \tag{2.4}$$

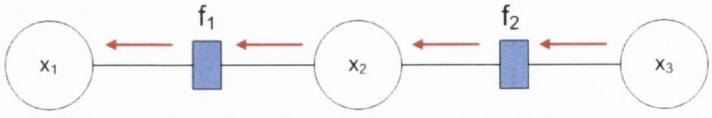
where, Z is partition function used for normalisation. For each variable vertex, local



(a) Factor graph



(b) Message passing in the forward direction.



(c) Message passing in the reverse direction.

Figure 2.2: Belief propagation using an example factor graph of three variable $X = \{x_1, x_2, x_3\}$ and two factor $F = \{f_1, f_2\}$ vertices.

marginal probability is computed using all messages passing through the vertex. There are two types of messages. The first type of messages pass from the variable vertices (x) to the factor vertices (f), i.e., $\mu_{x \to f}$. The second type of messages pass from the factor vertices (f) to the variable vertices (x), i.e., $\mu_{f \to x}$. For example, in the case of sum product messages passed in the forward direction are

$$\begin{aligned}
\mu_{x_1 \to f_1} &= 1 \\
\mu_{f_1 \to x_2} &= \sum f(x_1, x_2) \\
\mu_{x_2 \to f_2} &= \mu_{f_1 \to x_2} \\
\mu_{f_2 \to x_3} &= \sum f(x_2, x_3) \mu_{x_2 \to f_2}
\end{aligned}$$

and messages passed in the reverse direction are

$$\begin{aligned}
\mu_{x_3 \to f_2} &= 1 \\
\mu_{f_2 \to x_2} &= \sum f(x_2, x_3) \\
\mu_{x_2 \to f_1} &= \mu_{f_2 \to x_2} \\
\mu_{f_1 \to x_1} &= \sum f(x_1, x_2) \mu_{x_2 \to f_1}
\end{aligned}$$

in both directions the first messages are initialised with 1. $f(x_p, x_q)$ is a weight assigned to the edge connecting two neighbouring variable vertices. In the case of images, this function may be computed using properties such as colour, texture and brightness of the neighbouring pixels. Using these messages marginal probability of each vertex is computed which further

leads to assigning the best labelling to each variable vertex of the factor graph.

Belief propagation algorithms were initially developed for the simple graphs by Pearl [50]. Pearl showed that their algorithm for finding Maximum-a-Posteriori (MAP) assignments is guaranteed to converge and it will assign the optimal assignment values to each node. Simple graphs (without loops) used by Pearl were not able to capture complex pixel neighbourhood relations properly. Weiss and Freeman [51] extended the belief propagation algorithm for graphs of arbitrary topology.

In their work, Felzenszwalab and Huttenlocher [53] used a new message passing algorithms on multiscale grid graphs to reduce the computation time of basic loopy belief propagation algorithm. Their method captures long range pixel interaction by shorter paths at coarse levels. This information is further used at the higher levels to save computation time. Szeliski et al. [52] further improved computation time by introducing a new method for message passing.

2.3.2 Graph cut based methods

Greig et al. [55] were the first to show that graph cut (min-cut/max-flow) can be used to minimise energy function (Equation 2.1) for pixel labelling of a binary image. Their proposed framework has been extensively used in image and video segmentation [26, 20, 56, 28, 22, 57, 16, 44, 45]. Graph cuts can only work with certain kind of energy functions otherwise minimisation is NP hard. Greig et al. showed that the globally optimal solution of a two terminal, *st* graph (Section 3.2.1) can be achieved if the pairwise potentials are *semi-metric*. $\mathcal{D}_{p,q}$ is called a *semi-metric* if:

$$\mathcal{D}_{p,q}(\alpha, \beta) = 0 \Longleftrightarrow \alpha = \beta \tag{2.5}$$

$$\mathcal{D}_{p,q}(\alpha, \beta) = \mathcal{D}_{p,q}(\beta, \alpha) \geq 0 \tag{2.6}$$

where, α and β are labels. Boykov et al. [16] extended the framework of Greig et al. to minimise the energy function (Equation 2.1) for pixel labelling of colour or grey scale images. Their *move* making algorithms, the $\alpha\beta$-swap and the α-expansion are the most popular techniques for minimising the energy functions of form (Equation 2.1). The global minimum of a two terminal *st* graph can be computed using $\alpha\beta$-swap or α-expansion if pairwise potentials are *semi-metric* or *metric* respectively. $\mathcal{D}_{p,q}$ is called a *metric* if it satisfies

all 2.5 , 2.6 and

$$\mathcal{D}_{p,q}(\alpha,\beta) \leq \mathcal{D}_{p,q}(\alpha,\gamma) + \mathcal{D}_{p,q}(\gamma,\beta) \tag{2.7}$$

where, α, β and γ are labels.

The energy function (Equation 2.1) uses both pairwise and unary potentials to achieve favourable labelling. However, accuracy and the robustness of the achieved labelling can be improved with the use of higher order clique potential [48, 18, 17, 58]. Higher order clique potential takes into account three or more neighbouring pixels. In terms of the *Gibbs* energy, the higher order potential energy function can be written as:

$$E_{higher}(x) = \sum_{c \in \mathcal{C}} \mathcal{D}_c(x_c) \tag{2.8}$$

where term $\mathcal{D}_c(x_c)$ is potential function of the clique c, x_c is the set of all pixels in the clique c, i.e., x_c= $\{x_i, i \in \mathcal{C}\}$ and $\mathcal{C}$ is set of all cliques c. With this new potential the energy function can be defined as:

$$E(x) = \sum_{p \in \mathcal{P}} \mathcal{D}_p(x_p) + \sum_{\{p,q \in \mathcal{N}\}} \mathcal{D}_{p,q}(x_p, x_q) + \sum_{c \in \mathcal{C}} \mathcal{D}_c(x_c) \tag{2.9}$$

Kolmogorov and Zabin [48] were the first to minimise the certain form of higher order clique potential (clique of size 3 pixels) using graph cuts. In their method, optimal solution of an energy function E can be computed using a graph if and only if E is a submodular function. A function E is submodular if one of the following holds:

- E is a function of one variable.
- E is a function of two variables such that $E(A \bigcup B) + E(A \bigcap B) \geq E(A) + E(B)$.
- E is function of more than two variable and all projections of E on two variables are submodular.

Kolmogorov and Zabin added two additional nodes per clique to convert the higher order clique potentials into pairwise submodular potentials. Kohli et al. [17, 58] extended the framework proposed by Kolmogorov and Zabin. In their method, higher order clique potentials can be defined using any number of pixels. They showed that it is possible to achieve global minimum of a graph in polynomial time if higher order clique potentials are of a certain form $\mathcal{P}^n$ (Section 3.2.3).

2.3.3 Temporal consistency issues with label propagation based video segmentation methods

Despite current shortcomings (as mentioned below) label propagation based methods especially the α-expansion and the $\alpha\beta$-swap based algorithms produce better results than the block based video segmentation methods in terms of temporal consistency. Tracking based methods suffer badly in non textured areas (Section 2.2.1). Therefore, we focus on label propagation methods especially on the α-expansion to improve the temporal consistency of the state of the art of region based video segmentation methods. In chapter 3, we present a detailed analysis of the α-expansion algorithm along with our work.

We have identified following major areas when temporal consistency of segmentation can suffer. Our developed algorithms deals two out of three of these issues:

Comparing each pixel with each label of the learned label set: Label propagation methods should know the properties, e.g., colour and texture, of each label class in advance. Most of the methods learn these properties through an extensive training on test frames or previous frames. Outcome of training is a set of labels and their learned properties. Label propagation methods assign labels to each pixel p of frame I_t by comparing it with each label of the learned label set. In a video, generally a object can move only by a small distance in the next frame. In section 3.3, we have used this constraint in label propagation to produce more accurate and temporally consistent results with lesser number of pixel to label comparisons (Section 3.3).

Appearance of new objects: Label propagation uses the learned properties of the objects to assign labels to the pixels of current frame I_t. Hence, in the case of appearance of new objects in I_t, label propagation will assign incorrect labels to the pixels belonging to the new objects [17, 18]. In literature of label propagation based methods, we were able to find out only one approach, proposed by Wang et al. [49], for handling the appearance of new objects. Wang et al. [49] introduced a post processing method for handling the new objects. Their methods has three steps. After labelling each pixel of the current frame using α-expansion, in the first step their method measures χ^2 distance between colour distributions of *segments* at time t and $t-1$. Presence of a new object in any *segment* is detected if χ^2 distance is greater than a threshold value. In the next step, the mean-shift clustering algorithm is

used to cluster the pixels of the *segment* (where presence of new object is detected) in both the current (at time t) and the previous (at time $(t-1)$) frames. A comparison between clusters at time t and $t-1$ is used to identify pixels belonging to the new objects. In the last step, their method learn the property (i.e., colour distribution) of new objects and reruns the labelling algorithm within the *segments* where new objects were initially detected. The method proposed by Wang et al. has two big flaws. Firstly, their approach assumes that label propagation algorithm will assign the same label to the all pixels of a new object. However, this condition will fail in most of the cases. Secondly, the new object will not be detected if it is appearing very slowly in the video sequence. In section 3.6, we present a framework based on the energy minimisation to deal with appearance of new objects. Our method is inspired by the work of Wang et al. However, it addresses both limitations present in their method.

Region merge/split: As is the case with the block based video segmentation methods, temporal consistency of label propagation based methods suffer in cases of region merge/split. This issue is not addressed in our thesis.

2.4 Evaluation metrics

After segmenting a video sequence using either block or tracking or label propagation based methods, the question: How good is the segmentation method? needs to be answered. Performance of segmentation methods can be measured both analytically and empirically [59]. Methods belonging to the analytical group evaluate segmentation algorithms by considering their principles, requirements and complexity. The empirical methods evaluate segmentation algorithm by segmentation results. In this thesis, we used only the empirical methods for evaluating the performance of segmentation methods. For image/frame segmentation, empirical methods measure accuracy of segmentation results of each image/frame individually. This type of evaluation is called spatial accuracy evaluation. In video, segmentation results of sequential frames should not change significantly. Therefore, empirical methods should also measure the temporal consistency of segmentation results of each frame. Finally, performance of a segmentation algorithm for a video is computed using spatial accuracy and temporal consistency scores of all the frames.

Spatial accuracy and temporal consistency of segmentation results can be evaluated subjectively or objectively. Subjective evaluation is a qualitative evaluation where human observer rate the quality of segmentation results based on depth of their knowledge about the scene. European project COST211 first proposed a set of guidelines for segmentation quality assessment for human observer [60, 61]. These guidelines focus more on displaying the results. However, they do not specify any evaluation methodology. Later different approaches extended this framework by adding guidelines for evaluations [62]. Subjective evaluation is a time consuming process because it requires a large panel of human observers. On the other hand, objective evaluation is a fast way of evaluating the segmentation results quantitatively. However, findings from objective evaluation methods must align with the findings of subjective evaluation methods.

In sections 2.4.1 and 2.4.2, we present review of objective evaluation methods for measuring spatial accuracy and temporal consistency respectively.

2.4.1 Spatial accuracy metrics

Generally, spatial accuracy metrics work in two steps. In the first step, metrics identify pixels with incorrect label or error pixels in segmentation result of frames. In the second step, spatial accuracy score for frame is computed using scores of error pixels. Identifying error pixels is a trivial task if ground truth for each frame is available. The most simple and popular spatial accuracy metric was proposed by Wollborn and Mech [63] during the standardisation work of ISO/MPEG-4. This metric *Sqm* first uses ground truth for identifying the error pixels and then the total number of identified pixels are used as spatial accuracy score of the frame. *Sqm* gives equal weight to all the error pixels for the score computation.

Many researchers [64, 65, 66, 67] argued that weights should be given to pixels by taking their visual relevance into the account. Metric should penalize more if the error pixel belongs to visually relevant parts of the image. For example, Correia and Pereira [64] computed spatial accuracy in three steps. In the first step, individual segmentation quality for each object is measured by comparing spatial features, such as, area and perimeter of the segmentation results and the ground truth of each frame. In the second step, the relevance of each object is computed based on the visual attention (brightness and redness) it captures. Finally in the third step, spatial accuracy is obtained by combining these two measures. In another example, the method proposed by Marichal and Villegas [66] assign more weight to the error pixels

belonging to foreground objects in comparison to the error pixels belonging to background objects. Moreover, weight depends on distance of the error pixels from the object boundaries. For example, the error pixels belonging to boundary of a foreground object will be penalised more than the error pixels belonging to any background object but less than the error pixels present near the centre of a foreground object.

The scope of metrics which indicates only the number of visually relevant error pixels is very narrow because these metrics are not able to classify other types of errors present in the segmentation results. To overcome this limitation more than one metric is used to evaluate spatial accuracy. In their work Nascimento et. al [68, 69] used six simple metrics to classify the error into misclassification, detection failure, region merge (over-segmentation) and region split (under-segmentation). Over-segmentation is a process when a single object present in the ground truth gets segmented as several regions by the segmentation algorithm. Under-segmentation is a process when a single region present in the segmentation results represents more than one object of the ground truth. Region merge and region split metrics proposed by Nascimento et. al are very simple because they only indicate the presence or absence of over-segmentation and under-segmentation in segmentation results.

Gelasca et. al extended the work of measuring visual relevance [70, 62]. They proposed a complex framework for ranking six common causes of errors, or artefacts namely *added region, added background, inside holes, border holes* and *flickering* in the segmentation results based on their visual relevance perceived by human observers. *Added region* is a new region created due to over segmentation of the background. *Added background* is a part background attached to another object. *Inside holes* are under-segmented parts completely inside the object and *Border holes* are under-segmented parts which are attached to the border of the object. *Flickering* is the temporal variation of any of these artefacts. Their method used psychometric functions [71] for modelling the results achieved from human perception. Learned psychometric functions work as weights for the mentioned artefacts.

There are a small number of methods [72, 73, 74, 75] which do not use ground truth for measuring the spatial accuracy. These methods assess *goodness* of the segmentation results against some set of rules or assumptions. For example, method proposed by Erdem et al. [72, 73, 75] assumes that the object boundaries coincide with colour boundaries and motion boundaries. To measure the spatial accuracy of the segmentation results their method checks how well boundaries in the segmentation results coincide with actual boundaries of the objects

by taking the colour and the motion differences from both sides of the *segment's* boundaries. Higher scores will be achieved if boundaries in the segmentation results coincide with the actual boundaries. In another example, the method proposed by Correia and Pereira [74] used two *goodness* measures, intra-object and inter-object homogeneities for spatial accuracy. Intra-object homogeneity measure is composed of the shape regularity, spatial and motion uniformity. Inter-object disparity metric is composed of local colour and motion contrast with neighbours. Higher scores for both metrics means higher accuracy of the segmentation results.

In this thesis, we have developed three metrics, two metrics: *spatial consistency* and *size consistency* for measuring the spatial accuracy, and one metric *temporal consistency* for measuring the temporal consistency (Chapter 6). Our metrics for spatial accuracy are simple and use ground truth to detect the error pixels. These two spatial metrics are able to measure error caused by misclassification, over-segmentation and under-segmentation. These metrics are novel extension of the metrics proposed by Nascimento et. al [68, 69]. Their method only indicates the presence or absence of region split/merge in the segmentation results. Our metrics extend their work by penalising the segmentation results based on the number and the area of splitting/merging regions. However, this is not a significant change because region merge/split does not happen too often in a video sequence.

2.4.2 Temporal consistency metrics

Segmentation results of the objects should not change over the video sequence. However, if there is any change it should be consistent. Temporal consistency metrics should penalise any inconsistent change. Similar to the spatial consistency metric the most simple metric for measuring the temporal consistency was proposed by Wollborn and Mech [63] during the standardisation work of ISO/MPEG-4. Their temporal consistency metric, *Tqm* is estimated by taking a difference in the total number of error pixels (*Sqm*) in the current frame and in the previous frame. *Tqm* is very simple as it indicates the difference in the total number of error pixels. If the number of error pixels is same in the current frame and the previous frame *Tqm* will give high temporal consistency score to the segmentation results even though positions of error pixels are different.

The method proposed by Marichal and Villegas [66, 67] tries to overcome this limitation by taking the differences of false positives (pixels belonging to the object in segmentation result

but not in the ground truth) and false negatives (pixels belonging to the object in the ground truth but not in segmentation result) in the current frame and the previous frame separately. Moreover, any difference in distance between object's centroid in the segmentation result of the current frame and the previous frame with respect to the ground truth is used to penalise any change in the position of object over time [76, 77].

There are few approaches [74, 73, 75] which do not require ground truth for measuring the temporal consistency. These methods compute the temporal consistency of segmentation results of the current frame by measuring any change in properties, such as, colour histograms [75], area and elongation [64] of the objects from segmentation results of the previous frame. The underlying assumption is at least one property of objects will change if there is any change in the segmentation results. Therefore, more and more properties are needed to be included for improving temporal consistency computation.

Capturing only temporal consistency of segmentation results using region based properties (e.g., area, centroid, etc) of its objects is ambiguous. This is because these properties of objects can change due to both consistent or inconsistent segmentation results. For example, the area of object in the current segmentation results can change due to relative motion between objects and the camera or due to incorrect segmentation. These methods also cannot handle appearance, disappearance, merging and splitting of objects.

It is apparent from the literature that there is more focus on evaluating spatial accuracy of the segmentation results. There is a lack of robust approaches for measuring the temporal consistency of segmentation results without using ground truth. We have developed a novel algorithm to measure the temporal consistency by analysing the changes (e.g., shape and motion of *segments*) in segmentation results over time (more than one frame). For the physical objects moving with regular motions, if the rate of change is consistent over a period of time we infer that the segmentation results are temporally consistent. Our *temporal consistency* metric is successful in dealing occlusion, deocclusion, appearance, disappearance, merging and splitting of objects. The *temporal consistency* metric is one of the major contribution of this research work (Section 6.3).

Shape of a *segment* l at time t is represented by a $2D$ mask S_t of the same size of video frame where $S_t(x) = 1$ if pixel at position x is labelled as l in the segmentation result of frame I at time t. Shape mask has only the positional information of the segment. Any other information like the brightness, the colour and the texture information is not part of

the shape mask. Colour cues of the *segments* are modelled separately with the help of multi-dimensional Gaussian Mixture Model (GMM) where each dimension of GMM represent colour information of an individual colour channel (Section 3.4.1.1). Similarly, histograms of texture cues (*textons*) are used to represent the texture information (Section 3.4.1.2).

2.5 Conclusion

To date, accurate segmentation of video frames has received more attention than the consistent segmentation of video. A common belief is that an accurate spatial segmentation of video regions will automatically enforce the temporal consistency for regions in frames. However, consistency information can provide a major cue for video object detection and can also be used to improve video segmentation accuracy.

As discussed in section 2.1.4, most of the block based segmentation methods deliver short term temporal consistency. For long term temporal consistency, the employed segmentation method should aware of the future information like, temporal trajectories of all pixels of the objects for as far as possible in the video sequence, for segmentation of the current frame. The major problems with tracking based long term temporal consistency methods is the tracking of pixels or features in video sequences and poor performance in non textured regions. In comparison to both block based and tracking based methods, label propagation based methods produce better temporally consistent results. Therefore, in this thesis we have developed a novel approach based on the α-expansion for achieving robust temporally consistency results. In chapter 3, we discuss α-expansion algorithms in detail along with our novel contributions.

Empirical evaluation methods should measure the spatial accuracy and the temporal consistency for evaluating video segmentation results. Spatial accuracy metrics measure the goodness of frame/image segmentation with the help of either ground truth or image properties, e.g., colour homogeneity and boundaries. Temporal consistency metrics try to measure variations in segmentation results over time. In literature, there is more focus on the development of spatial accuracy metrics. There is a clear lack of robust algorithms for measuring temporal consistency. In chapter 6, we present our three metrics for measuring spatial accuracy and temporal consistency. Our spatial consistency metrics measure the accuracy of segmentation results of frames using human annotated ground truth. Our temporal consistency metric does not require ground truth.

Chapter 3

Label propagation in forward direction

Video segmentation can be seen as a label propagation problem where the labels existing in previous frames are propagated to assign a label to each pixel p of the current frame I_t. Label propagation based methods minimise a defined energy function to achieve the best labelling. In this thesis, we define our energy function using unary, pairwise and higher order clique potentials (Section 3.1). The α-expansion and the $\alpha\beta$-swap are the two most popular *move* algorithms for energy minimisation [16] (Section 3.2). After starting from an initial labelling both algorithms change (*move*) the labelling iteratively. A change which leads toward the minimum energy is selected at each iteration. Final labelling is said to be achieved when further minimisation of energy is not possible. We have used α-expansion to minimise our energy function (section 3.4). In section 2.3.3, we identified three major issues with label propagation based methods:

1. Each pixel is compared with all labels present in the set of labels.
2. There is no mechanism for handling appearance of new objects.
3. There is no procedure to use region merge/split information happening in the future in the current frame.

In this chapter, we present algorithms to deal with first two issues:

- **Spatio temporal displacement constraint :** We propose a novel method that enables label propagation methods to produce robust results with lesser pixel to label comparisons. In our method, each pixel is compared with a small number of labels present in its neighbourhood in the previous frame. Another advantage of our method is that it can handle *non class based* segmentation (Section 3.3).

- **Incorporating new objects in label propagation:** A preprocessing step is developed for detecting pixels belonging to new objects (Section 3.6).

Label propagation results of our method are presented in section 3.5.

3.1 Defining energy function

Our energy equation with unary potential, pairwise potential and higher order clique potential for labelling the current frame is defined as:

$$E(x) = \sum_{p \in \mathcal{P}} \mathcal{D}_p(x_p) + \sum_{\{p,q \in \mathcal{N}\}} \mathcal{D}_{p,q}(x_p, x_q) + \sum_{c \in \mathcal{C}} \mathcal{D}_c(x_c) \tag{3.1}$$

where, $\mathcal{D}_p(x_p)$ is a function of assigning label x_p to any pixel $p \in \mathcal{P}$ and $\mathcal{D}_{p,q}(x_p, x_q)$ measures the cost of assigning labels x_p and x_q to pixels p and q, respectively. $\mathcal{D}_{p,q}$ term encourages spatial smoothness by assigning same label to both the pixels $\{p, q\}$. $\mathcal{D}_c(x_c)$ is a potential function of the clique c, x_c is the set of all pixels in the clique c and $\mathcal{C}$ is set of all cliques (Section 3.4.3). This term encourages smoothness by assigning same labels to all the pixels of clique.

Equation 3.1 computes cost of assigning labelling x to all pixels of image by adding the sum of unary costs $\mathcal{D}_p$ of all pixels, the sum of pairwise costs $\mathcal{D}_{p,q}$ of all neighbouring pixel pairs and the sum of higher order clique costs $\mathcal{D}_c$ of all cliques. An optimum labelling is achieved if total energy $E(x)$ is minimum.

3.2 Optimising energy function with α-expansion for label propagation

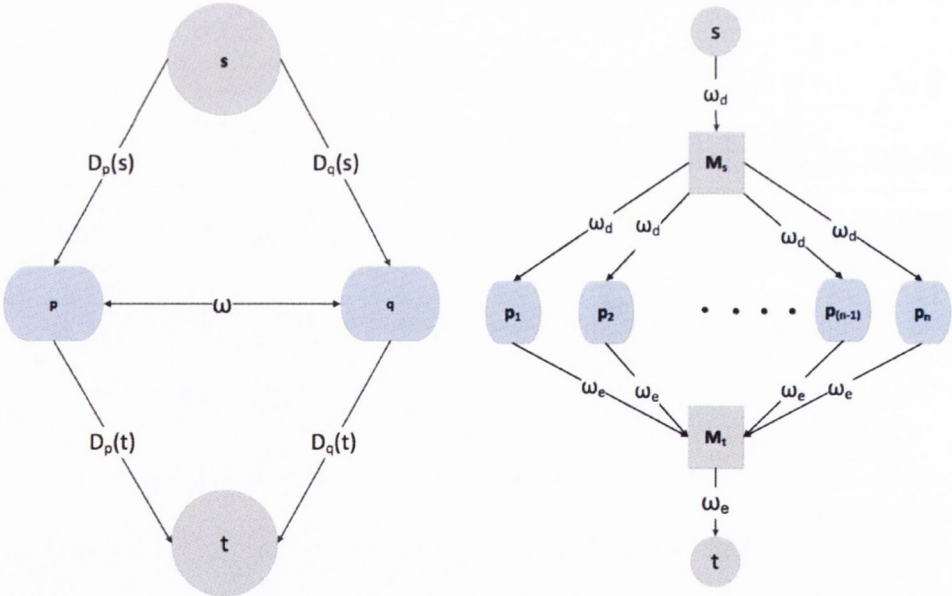
Boykov et al. [16] proposed the α-expansion *move* algorithm based on graph cuts to minimise the energy function. The algorithm starts with an initial labelling x and change labels of a large set of pixels at each iteration to lower the labelling energy $E(x)$. A best labelling x is said to be achieved for which further minimisation of $E(x)$ is not possible. In our research, we have used α-expansion for minimising our energy function $E(x)$ (Equation 3.1). In this section, we review the basics of minimising energy function in the following steps:

- two terminal *st* graph construction (Section 3.2.1),

- and minimisation of pairwise (Section 3.2.2) and higher order (Section 3.2.3) potentials of energy function $E(x)$.

3.2.1 *st* graph

Move algorithms create a two terminal directed weighted graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges. In order to minimise the energy equation with only the unary and the pairwise potential terms the node set $\mathcal{V}$ holds all pixels or features points or *superpixels* and two additional nodes: source s and sink t. The edge set $\mathcal{E}$ holds two type of edges, t-link and n-link. t-links connect nodes with source and sink terminals and n-links connect neighbouring nodes. Weights are associated with all the edges in graph. The weight of a t-link is the cost of assigning label of the terminal to the node. This weight is calculated from the unary term D_p. Similarly the weight of a n-link connecting two nodes p and q is the smoothness cost $D_{p,q}$ (Figure 3.1a).



(a) Unary and pairwise potentials

(b) Higher order clique potential

Figure 3.1: *st* graph for computing the optimal moves for the $\mathcal{P}^2$ [78, 16] and the $\mathcal{P}^n$ [48, 17] Potts models. Two graphs are shown (a) for a single neighbourhood and (b) for a single clique. Additivity theorem [48] allows us to merge these graphs to construct a single *st* graph for minimising all three unary, pairwise and higher order clique potentials.

Kolmogorov and Zabin [48] showed that along with pairwise potentials, α-expansion can also be used to minimise higher (of size 3) order clique potentials $\mathcal{D}_c$, if $\mathcal{D}_c$ is a submodular

function (Section 2.3.2). They added two additional vertexes called *auxiliary* vertexes, M_s and M_t in the *st* graph $\mathcal{G}$ to represent a higher order clique (Figure 3.1b). Moreover, these auxiliary vertexes help to create projections of $\mathcal{D}_c$ on functions of two variables (pairwise potential).

An *st cut* partitions the nodes $\mathcal{V}$ of a *st* graph into two disjoint subsets $\mathcal{S}$ and $\mathcal{T}$. Nodes present in $\mathcal{S}$ subset will be labelled as $\mathcal{S}$ and nodes of subset $\mathcal{T}$ will be labelled as $\mathcal{T}$. The cost of a *st cut* $\mathcal{C} = \{\mathcal{S}, \mathcal{T}\}$ is defined as the sum of weights of edges (p, q) where $p \in \mathcal{S}$ and $q \in \mathcal{T}$. A *mincut* is the cut which have minimum cost $\mathcal{C}$. In graph theory, edges present in an *st cut* are also called saturated if no more flow is possible between the source and the sink through these edges.

Figure 3.1, shows two example *st* graphs for computing the optimal moves for the $\mathcal{P}^2$ [78, 16] and the $\mathcal{P}^n$ [48, 17] Potts models. These graphs are created for a single neighbourhood (Figure 3.1a) and for a single higher order clique (Figure 3.1b). Figure 3.1a, shows two neighbouring pixels p and q with the source s and the sink t terminals. Four weighted t-links connect p and q with the source s and the sink t terminals where t-links weights $D_p(s)$, $D_p(t)$, $D_q(s)$ and $D_q(t)$ are the unary costs for assigning the labels of the source and the sink terminals to the pixels p and q respectively. Pixels p and q are also connected with each other using a weighted n-link. In figure 3.1b, a higher order clique is created using pixels $\{p_1, p_2 \dots p_n\}$. Two additional vertexes M_s and M_t are added to covert all pairwise projections of the higher order clique into submodular functions. Pixels $\{p_1, p_2 \dots p_n\}$ are connected with M_s and M_t using weighted t-links. The minimisation process of these graphs are detailed in sections 3.2.2 and 3.2.3.

3.2.2 Minimising pairwise potential using α-expansion

Boykov et al. [16] illustrated that an energy function consisting of unary and pairwise clique potentials can be minimised using the α-expansion, if the pairwise clique potential takes the form of $\mathcal{P}^2$ Potts model. Potts model $\mathcal{P}^2_{p,q}$ can be written as:

$$\mathcal{P}^2_{p,q} = \begin{cases} \gamma_1 & if\ x_p \neq x_q \\ \gamma_2 & else \end{cases} \tag{3.2}$$

where, both γ_1 and γ_2 are non negative. If $\gamma_k = 0$, this function becomes a metric function and hence can be minimised using α-expansion algorithm. Unary potential is a function of

one variable and is always metric (Equation 2.7). Therefore it does not require any specific model for minimisation.

The α-expansion algorithm is an iterative algorithm where in each *move*/iteration it allows any set of pixels to either retain their current labels or be assigned a label $\alpha \in \mathcal{L}$ where $\mathcal{L}$ is the set of learned labels. The labelling returned after any such move t_p can be represented as:

$$x'_p = \begin{cases} x_p & if\ t_p = 0 \\ \alpha & if\ t_p = 1 \end{cases} \tag{3.3}$$

At each iteration, for each $\alpha \in \mathcal{L}$, there are many possible *moves* (change in labelling of pixels). Boykov et al. [16] showed that, an optimal labelling $\hat{x}$, i.e., $\hat{x} = \arg\min E(x')$ can be computed using α-expansion if $\mathcal{D}_{p,q}$ is of form of $\mathcal{P}^2_{p,q}$. After α-expansion pairwise potential can be expressed as:

$$\mathcal{D}_{p,q} = \begin{cases} \gamma & if\ x_p \neq x_q \\ 0 & else \end{cases} \tag{3.4}$$

where, γ is non negative.

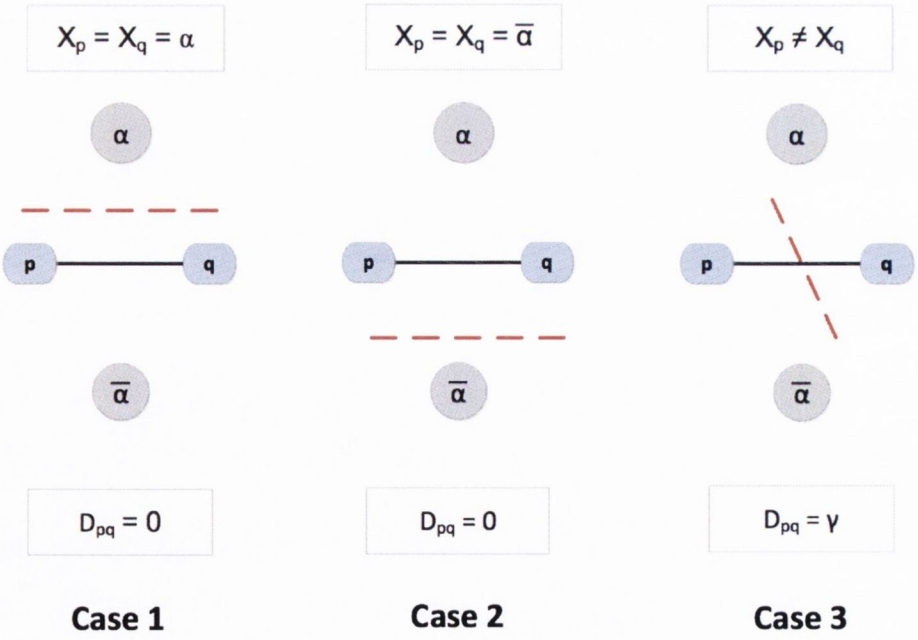


Figure 3.2: *st* graph and pairwise potentials after α-expansion. Red lines represent the *st-mincut* and black lines are graph edges.

Any pairwise potential function can be converted into metric form with the help of Potts

model and hence can be minimised using α-expansion. In order to show this, we have considered three cases (Figure 3.2):

- Cases 1 and 2: In both cases, the same label is assigned to nodes p and q. Therefore, the *st-mincut* does not cut the edge connecting nodes p and q. According to equation 3.4, the cost of the pairwise potential $\mathcal{D}_{pq}$ will be equal to zero.
- Case 3: In this case, different labels are assigned to nodes p and q. Therefore, the *st-mincut* cuts the edge connecting nodes p and q. According to equation 3.4, the cost of the pairwise potential $\mathcal{D}_{pq}$ will be equal to γ.

3.2.3 Minimising higher order clique potentials

Kohli et al. [17] introduced the $\mathcal{P}^n$ Potts model family for the higher order clique potentials. They proposed that the higher order clique potential can be minimised using α-expansion algorithm, if it takes the form of $\mathcal{P}^n$ Potts model. Their $\mathcal{P}^n$ Potts model defined for a clique c of size n is a generalisation of pairwise Potts model family:

$$\mathcal{D}_c(x_c) = \begin{cases} \gamma_k & if\ x_i = l_k \forall i \in c \\ \gamma_{max} & else \end{cases} \tag{3.5}$$

where, $\gamma_k \geq 0$ and $\gamma_{max} \geq \gamma_k, \forall l_k \in \mathcal{L}$.

The higher order clique potential function that takes the form of $\mathcal{P}^n$ Potts model can be given as:

$$\mathcal{D}_c(x_c) = \begin{cases} \gamma_\alpha & if\ x_p = \alpha\ \forall p \in c \\ \gamma_{\bar{\alpha}} & if\ x_p = \bar{\alpha}\ \forall p \in c \\ \gamma_{max} & otherwise \end{cases} \tag{3.6}$$

where, $\gamma_{max} \geq \gamma_\alpha$ and $\gamma_{max} \geq \gamma_{\bar{\alpha}}$.

Figure 3.1b shows the graph constructed corresponding to the values of the above potential for a single clique. Two auxiliary nodes M_s and M_t are added into the graph to create the projections of higher order clique potential on two variables. Moreover, the weights of the edges are positive and given by $w_d = \gamma_{\bar{\alpha}} + k$ and $w_e = \gamma_\alpha + k$. Kohli et al. [17] used the following three cases to show that the graph constructed in figure 3.1b corresponds to the clique potential plus a constant in equation 3.6:

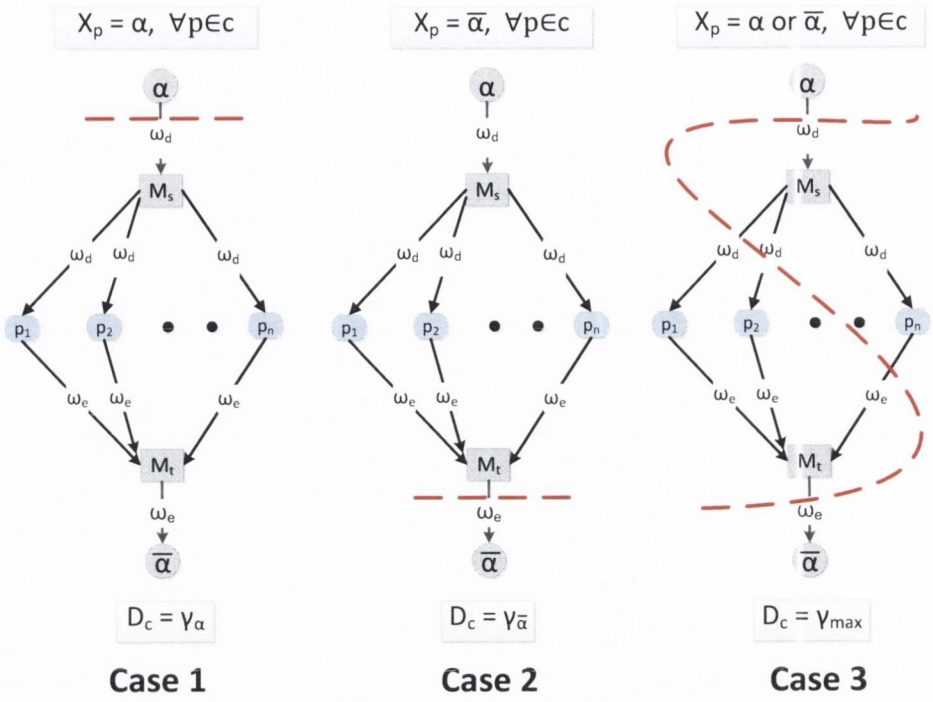


Figure 3.3: *st* graph and higher order clique potentials after α-expansion. Red lines represent the *st-mincut* and black lines are graph edges (a) both pixels got label α (b) both pixels got label β (c) both pixels got different labels [17].

- Case 1: Label of the source terminal α is assigned to all the pixels belonging to the clique. Therefore, the *st-mincut* corresponds to the edge connecting M_s to the source which has a cost $w_d = \gamma_\alpha + k$. (case 1 in figure 3.3).

- Case 2: Labels of the pixels remain unchanged. Therefore, the *st-mincut* corresponds to the edge connecting M_t to the source which has a cost $w_e = \gamma_{\bar{\alpha}} + k$ (case 2 in figure 3.3).

- Case 3: For all remaining cases, the *st-mincut* corresponds to the edges connecting M_s to the source and M_t to the sink terminal. The cost of the cut is $\gamma_{max} + k$ where $\gamma_{max} = w_d + w_e + k$ (case 3 in figure 3.3).

3.3 Spatio temporal displacement constraint

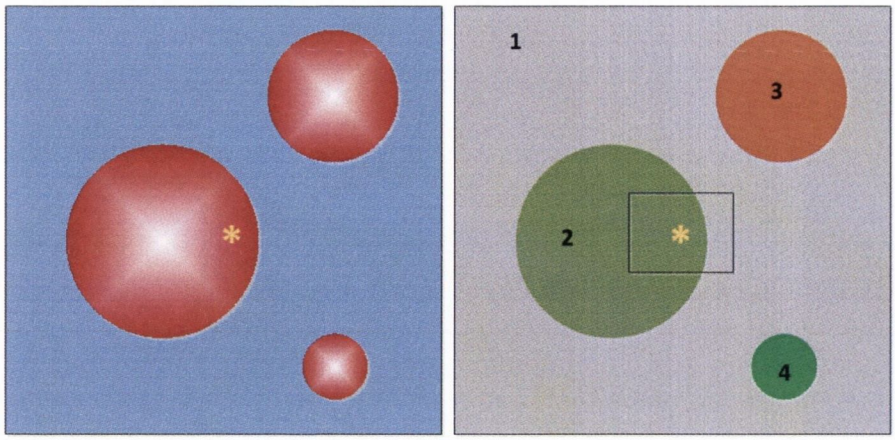
In practise, each label in label propagation represents a class of physical objects like sky, car, grass, road, boat and sheep. We call this type of label propagation a *physical class based* label propagation. In the *physical class based* label propagation methods, each label class

has *unique* properties, e.g., colour and texture, and represents a class of physical objects. Therefore, *segments* of a video sequence represent physical objects and in any frame, multiple instances (e.g., presence of many cows) of the same object class gets the same label. If different labels are used to represent multiple instances of a physical class, then it may be referred to as *non class based* label propagation. The label propagation process should know or learn unique properties of each label in advance. Every pixel of a frame gets a label l of the most similar class present in the learned label set $\mathcal{L}$. For example, there are two label classes cow and grass present in figure 3.4 and any *physical class based* label propagation which knows the properties (colour and texture distributions) of both cow and grass classes can be used for segmentation.

Figure 3.4: Physical class based segmentation, figure from [23]. Labels are propagated considering three conditions; colour distance between pixel and class, spatial coherence among pixels and higher order co-occurrence statistics.

Physical class based segmentation methods assign a label to any pixel p of frame I_t at time t after comparing it with each label l of learned label set $\mathcal{L}_{t-1}$ present at time $t-1$. In our approach, *segments* may represent instances of same physical classes so it is possible to have multiple labels with *similar* properties (*non class based* label propagation). Therefore, we should not compare each pixel with each label present in the label set. Any pixel p of I_t should only be compared with a subset of labels $l \in \mathcal{L}^p_{t-1}$ where $\mathcal{L}^p_{t-1} \subseteq \mathcal{L}_{t-1}$ based on physical proximity.

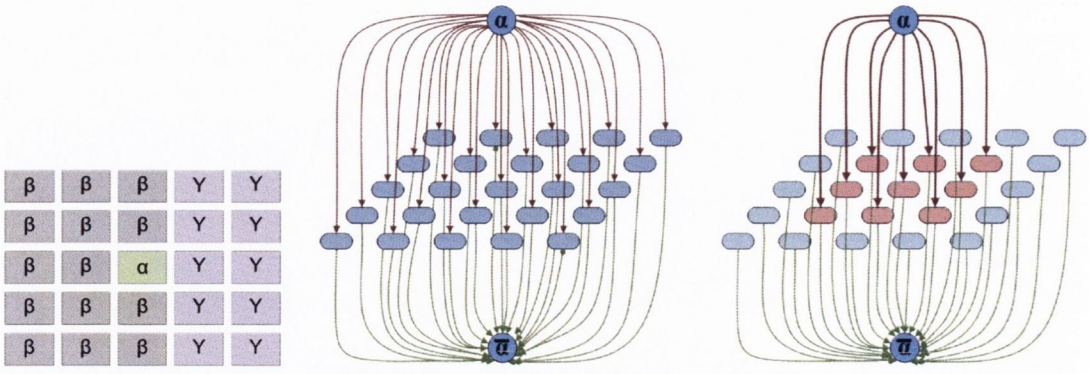
In a video sequence, if there is no sudden camera movement, any object present in frame I_{t-1} can only move by a small distance in the next frame I_t. We refer this constraint as the *spatio temporal displacement constraint* and use it to determine the label subset $\mathcal{L}^p_{t-1}$ to be considered for each pixel p. In figure 3.5a, there are three instances of a class *ball* present in a frame I_t, and by our method, segmentation result (label mask) of frame I_t should have



(a) (b)

Figure 3.5: Temporal neighbourhood of pixel p holds only a small subset of labels $\mathcal{L}_{t-1}^{p} = \{1, 2\}$ for d_{max}. (a) Location of pixel p is shown as '*' in I_t. (b) placing a mask of $[(2d_{max} + 1) \times (2d_{max} + 1)]$ dimension around p in the segmentation results of frame I_{t-1}.

different labels for all three instances. Any physical class based label propagation methods will assign same label to all three balls. In our label propagation method, each pixel p of I_t will be compared with a subset of label $\mathcal{L}_{t-1}^{p}$ where $\mathcal{L}_{t-1}^{p}$ has only labels of the *segments* present within d_{max} pixel distance from p in the segmentation mask of frame I_{t-1}. d_{max} is the maximum displacement (in pixels) possible between any two successive frames I_t and I_{t-1}. For any pixel p (p_x, p_y), a window of $(2d_{max} + 1) \times (2d_{max} + 1)$ dimension is placed at the coordinates (p_x, p_y) in the label mask of the frame I_{t-1} (Figure 3.5b). Labels present in window become a member of the $\mathcal{L}_{t-1}^{p}$ subset. In figure, 3.5b, a subset of segments $\mathcal{L}_{t-1}^{p} = \{1, 2\}$ is present in the spatio temporal neighbourhood of p.



(a) Label mask of I_{t-1} (b) Without spatio temporal displacement constraint (c) With spatio temporal displacement constraint

Figure 3.6: *st* graph construction for frame I_t of size 5×5. In both cases, graphs are only showing t-links with unary potentials (data term) as weights. Two additional vertexes added to incorporate higher order clique potentials are also not shown in the figure.

In α-expansion based label propagation methods, any label $\alpha \in \mathcal{L}$ can be assigned to any pixel p of I_t. Hence all pixels of I_t are required to be connected with both terminals of st-graph through edges called t-links. In our algorithm, a label $\alpha \in \mathcal{L}$ can be assigned only to a small number of pixels of I_t. Therefore, only pixels p where $\alpha \in \mathcal{L}_{t-1}^{p}$ are required to be connected with the source (α) terminal. This condition reduces the number of t-links in st graph significantly. In the other words, using our algorithm any label α can be propagated with significantly fewer pixel to label comparisons. For example, figure 3.6 shows st graphs created for propagation of a label, α for a 5×5 frame with and without spatio-temporal displacement constraint. Only t links are shown in st graphs. Figure 3.6a shows label mask of the previous frame where labels α, β and γ are assigned to different pixels. Figure 3.6b shows the st graph created for propagating label α in the current frame using α-expansion based methods. In α-expansion based methods, any label α, β, γ can be assigned to the pixels of the current frame. Therefore, all pixels are connected with the source (α) and the sink ($\bar{\alpha}$) nodes. In our method, label α can only be assigned to pixels for which α is present in the spatio-temporal neighbourhood, these pixels are highlighted with brown colour in figure 3.6c. As a result, source (α) node is only connected with pixels highlighted with brown colour in the st graph. In this illustration, we used $d_{max} = 1$.

Spatio-temporal displacement constraint is one of the main contributions of this thesis. It allows α-expansion method to achieve labelling with lesser number of comparisons between pixels and labels during each iteration. We have used meta data, e.g., size of objects present in ground truth and frame per second to select the optimal value of d_{max}. For example, d_{max} value for videos with higher frame rate will be smaller than d_{max} value for videos with lower frame rate. Pixels will be misclassified if any object moves by more than d_{max} distance in the sequential frames.

3.4 Energy potentials with spatio temporal displacement constraint

In this section, we discuss methods for computing the unary (Section 3.4.1), the pairwise (Section 3.4.2) and the higher order clique (Section 3.4.3) potentials. In each section, we also present our method for computing the energy potentials with the spatio temporal displacement constraint.

3.4.1 Unary potential

For label propagation in frame I_t, for each pixel p of the current frame I_t, the unary potential should be able to capture the disagreement between a pixel and any of the label classes and it should encourage assignment of label of the most similar label class to the pixel. Generally, colour features have been used to define the unary term [11]. However, colour alone is not very deterministic and hence in this research texture features are used with colour features to define the unary potential. Using both colour and texture the unary potential can be defined as:

$$\mathcal{D}_p(x_p) = \delta_{\{p,x_p\}} * (\theta_{col} * \mathcal{D}_p^{col}(x_p) + \theta_{tex} * \mathcal{D}_p^{tex}(x_p)) \tag{3.7}$$

and

$$\delta_{\{p,x_p\}} = \begin{cases} 1 & if\ x_p = l\ and\ l\ \in \mathcal{L}_{t-1}^p \\ \infty & otherwise \end{cases}$$

where, θ_{col} and θ_{tex} are weighting parameters for colour potential $\mathcal{D}_p^{col}(x_p)$ (Section 3.4.1.1) and texture potentials $\mathcal{D}_p^{tex}(x_p)$ (Section 3.4.1.2) respectively. These weights are selected empirically ($[0, 1]$) depending on the nature of undertaken videos. For example, value of θ_{tex} will be higher than the value of θ_{col} for texture rich videos. In section 3.3, we argued that during the label propagation any pixel p of the current frame should only be compared with the labels present in a small subset $\mathcal{L}^p$ rather than every label present in the learned label set $\mathcal{L}$ where $\mathcal{L}^p \subseteq \mathcal{L}$. In order to enforce this spatio temporal displacement constraint, our unary potential is computed by multiplying the unary potential in equation 3.7 with a function δ. This is similar to applying a local pillar box filter to a function.

In equation 3.7, cost of assigning any label $l = x_p$ to a pixel p which is not present in the spatio temporal neighbourhood of p is infinity. Therefore, move algorithms will avoid such assignments. For each iteration of the α-expansion, on the *st* graph, an edge between a pixel and a terminal is only created if the edge weight is not infinity.

In equation 3.7, weighted sum of colour and texture potentials is used as unary potential. In an alternative approach, this potential could be computed in a joint feature space created by adding both features. For example, for each pixel the colour feature vector of length 3 and texture feature vector of length 17 could have been merged to create a joint colour-texture

vector of length 20. However, adding cue potential separately gives equation more flexibility. Weights can be assigned to cues depending on the video. For example, in segmentation process a video with rich texture information texture cue should play the important role. Thus, more weight should be assigned to the texture cue.

3.4.1.1 Colour potential

The Gaussian Mixture Model (GMM) is a common choice to model the colour information [49, 79] of labels where Expectation Maximisation (EM) is used to learn the parameters of mixture model from training data. For a pixel p of the current frame I_t, our colour potential is computed as the negative log likelihood of the probability P_g of assigning a label $l = x_p$ to p as:

$$\begin{aligned} \mathcal{D}_p^{col}(x_p) &= -logP_g(I_t(p)|x_p; \Theta_{col}) \\ P_g(I_t(p)|x_p = l; \Theta_{col}) &= \sum_{k=1}^{K_n} w_{lk} \mathcal{N}(I_t(p); \mu_{lk}, \sum_{lk}) \end{aligned} \tag{3.8}$$

where, w_{lk}, μ_{lk} and $\sum_{lk}$ are the weight, the mean and the covariance parameters (Θ_{col}) of k^{th} component of GMM, i.e., $k = 1, \ldots K_n$. $P_g(I_t(p)|x_p; \Theta_{col})$ measures the similarity between the colour of a pixel p and the colour of a *segment* l. A higher value is expected when pixel colour is similar to the colour of *segment*. Equation 3.1 should be minimised for achieving the best labelling. For this purpose, negative logarithmic of P_g is taken as unary cost $\mathcal{D}_p^{col}$. $-log(P_g)$ assigns a smaller value to $\mathcal{D}_p^{col}$ for the best colour match. Thus, encouraging the minimisation algorithm for assigning the label of the *segment* with the best colour match to the pixel.

Learning parameters of any parametric model, e.g., Gaussian Mixture Model, requires training. Therefore many authors prefer non-parametric models to compute colour potentials. Histograms [14] and kernel density estimation [80] are two examples of the non parametric methods. For the approaches described in this thesis, video frames are represented in CIE Lab colour space. International Commission on Illumination specify CIE Lab is the most complete colour space. This colour space describes all the colours visible to human eyes and distance between any two colours is directly proportional to the difference between the two colours as perceived by human eyes. The L channel of CIE Lab represents lightness of colour on scale

of 0 and 100 where 0 represents black and 100 represents white. The A channel represents colour value between pure green and pure red where value of pure green is represented by -127 and value of pure red is represented by $+127$. The B channel represents the colour value between pure blue and pure yellow where -127 represents value of pure blue and $+127$ represents value of pure yellow [81]. We have also tested the RGB colour space but better segmentation results are achieved in the CIE Lab colour space.

3.4.1.2 Texture potential

In an image, variations of properties such as intensities, colours and shapes which form a repeated pattern is called visual texture. Generally, computer vision methods determine texture cues in two steps. In the first step, methods extract the underlying texture information from images and in the next step texture cues are defined using the extracted information. Tuceryan and Jain [82] divided the methods of texture analysis into four categories: statistical, geometrical, model based and signal processing methods.

Statistical methods describe texture using statistical measures. Grey level co-occurrence matrix (GLCM) introduced by Haralick [83] is one of the most well known and commonly used statistical method for determining texture information. These matrices store frequencies of grey level pairs of pixels at a certain distance d. If image has G grey levels, size of GLCM matrix will be $G \times G$. Any entry $GLCM_{(i,j)}$ in GLCM is the number of occurrence of the pair of grey levels i and j which are a distance d apart in the image. Finally, the statistical properties, e.g., entropy, correlation, homogeneity, etc., extracted from the co-occurrence matrix are used as texture features. Quality of texture features highly depends on the distance d. However, there is a lack of robust methods for the selection of the optimal d.

Geometrical methods assume that textures is produced by placing of texture primitives like Voronoi polygons [84] and blobs [85, 86] according to a certain placement rule. Generally, these methods determine underlying texture primitives by applying a set of filters like Laplacian of Gaussian (LoG) to the image. The placement rule is further defined with the help of graphs. This class of methods can only deal with the regular textures due to requirement of a placement rule for defining the texture.

Model based methods infer the texture information by representing the image using models like Markov random field (MRF) [87, 88] and Fractals [89]. Markov random fields capture the local information by assuming that properties such as intensity and colour of

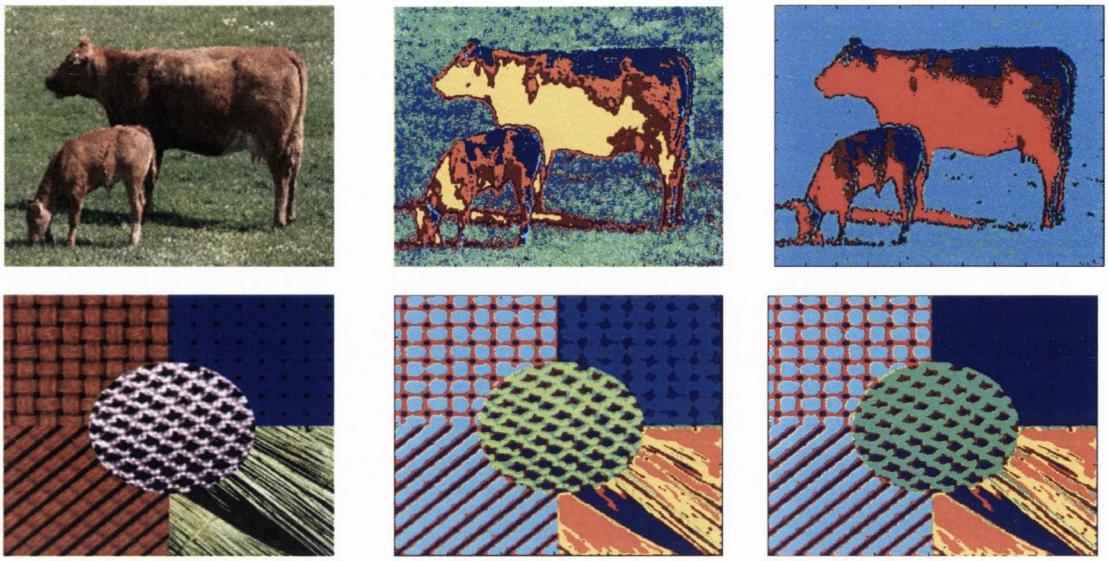
each pixel in the image depend only on the properties of the neighbouring pixels. MRF based models define texture properties of an image by estimating the parameters of underlying model.

In natural surfaces texture may present at different scales. Markov random fields are only able to capture the local information. Therefore, models like fractals which can measure texture at different scales are used. A fractal is defined using the concept of self-similarity. A bounded set A in Euclidean of n-space is said to self similar when A is the union of N non-overlapping and distinct copies of a scaled down (by factor r) version of itself. Properties of fractal like its dimension [90] and lacunarity [91] are further used to define image texture.

Signal processing methods represent texture using frequency of image properties such as number of edges present in per unit area. A sliding window provide a direct approach to capture image properties. In sliding window approach, texture cues for each pixel (central pixel) are generated by sliding a window across the image. This approach assumes that a window centred around a pixel can capture its texture using properties of pixels present inside window [17, 92]. However, this assumption is very sensitive to noise and can fail near the object boundaries. Filters in spatial and frequency domains provide a robust way of capturing image properties [93, 18]. Similar to geometrical methods a set of filters are applied to the image to capture texture properties. The resulting texture information is robust to noise and to any geometrical transformation, such as rotation and scaling. Therefore, we have adopted this approach for capturing the texture information.

The filter bank used in our method was proposed by Winn et al [94]. This filter bank consists of 17 filters namely three Gaussian (with σ = 1, 2, 4), four Laplacian of Gaussian (with σ = 1, 2, 4, 8), two first order derivative of Gaussians in X dimension (with σ = 2, 4) and two first order derivative of Gaussians in Y dimension (with σ = 2, 4) filters. The three Gaussian filters (with σ = 1, 2, 4) are applied to each channel of CIE LAB image, the Laplacian of Gaussian (with σ = 1, 2, 4, 8) and the first order derivative of Gaussians filters (with σ = 2, 4) are applied to only L channel. Convolution of the frames with the filter bank, transforms each image pixel into a multi-dimensional feature vector. These resulting filter responses are highly redundant. Therefore, clustering approaches are applied to separate the dominant responses. These dominant filter responses are called *textons* and associated filter response vectors are called *appearance vectors* [95].

The K-means algorithm is used to cluster these filter responses. Generally, K-means re-



(a) (b) (c)

Figure 3.7: Texture map (a) Original image (b) initial clusters using K-means (k = 10) (c) cluster merging using Mahalanobis distance.

quires prior information about the number of clusters. Final results also depends on starting cluster origins. Therefore, K-means is applied in two steps. In the first step, K-means algorithm is initialised with the a large number of clusters (1000 in our case). This step is repeated multiple times (maximum 100 iterations in our case) with different starting points and the solution where sum of the distances between points and their cluster centres is minimum, gets selected for the next step. In the second step, number of K-mean cluster reduced by combining statistically similar initial clusters [96]. In our case, the *Mahalanobis* distance is used to measure statistical similarity between any two clusters i and j:

$$Dist(i,j) = \sqrt{(\mu_i - \mu_j)^T \left[\sum_i + \sum_j\right]^{-1} (\mu_i - \mu_j)} \tag{3.9}$$

where, μ_i, μ_j are the mean vectors and $\sum_i, \sum_j$ are the covariances of the feature vectors of regions i and j. Mahalanobis distance is preferred over Euclidean distance because it takes covariances into account, leading to more accurate cluster boundaries. Textons returned by K-means represent the dominant local textures present in the training image. Figure 3.7 shows generated texture maps $\mathcal{T}$ for different images using the described method.

Finally, a normalised histogram of *textons* $\mathcal{H}$ of pixels of each *segment* is used to define texture of the *segments*. $\mathcal{H}$ is created for each segment using the manually specified segmentation for the first processed frame. For a pixel p of current frame I_t, the texture potential is

computed as the negative log likelihood of the probability P_g of assigning a label l to p:

$$\begin{aligned} \mathcal{D}_p^{tex}(x_p) &= -logP_g(\mathcal{T}_t(i)|x_p;\Theta_{tex}) \\ P_g(\mathcal{T}_t(i)|x_p = l;\Theta_{tex}) &= \mathcal{H}^l(\mathcal{T}_t(i)) \end{aligned} \tag{3.10}$$

In equation 3.10, P_g is computed using the normalized texton histogram $\mathcal{H}^l$ learned from the texton map $\mathcal{T}^l$ of label l in the first (training) frame.

Texton maps are similar to the image representation in the S-CIE Lab colour space. S-CIE Lab is a spatial extension of the CIE Lab colour space with a spatial filtering pre-processing step. The filtering is performed in an opponent colour space, containing one luminance and two chrominance channels. Opponent colour space is a linear transformation of CIE 1931 XYZ or LMS colour spaces. Generally, two dimensional Gaussian filters are used as spatial filters and a weighted sum function is used to add all the filter responses for each pixel position. The spatial filters in S-CIE Lab colour space approximates the contrast sensitivity functions of the human vision system [97]. In S-CIE Lab colour space, each pixel position is a three dimensional vector of filtered colours. However, each *appearance* vector in texton map has several dimensions (17 in our case) capturing different features like number of edges per unit area of the image.

3.4.2 Pairwise potential

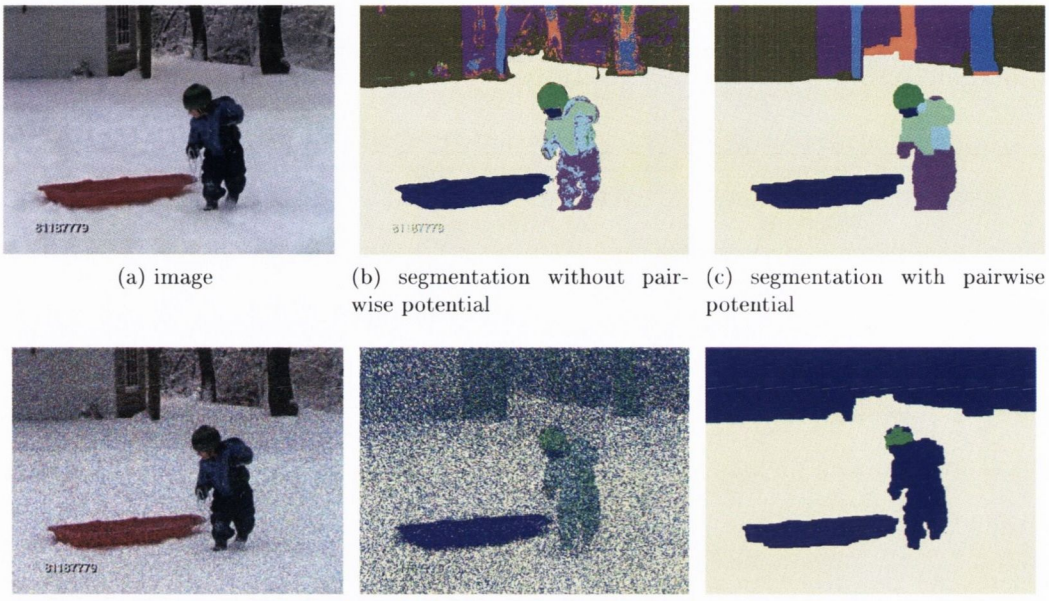
The spatial smoothness term encourages continuity of segment labelling $\mathcal{L}_t$ in frame I_t. Segment boundaries will suffer if the label propagation method tries to enforce smoothness everywhere during label propagation. Therefore, spatial smoothness terms should be discontinuity respecting. Spatial smoothness can be interpreted as a discontinuity penalty between neighbouring pixels, $p, q \in \mathcal{N}$, where $\mathcal{N}$ is a set of pairs of 4-connected neighbouring pixels and value of spatial smoothness depends on intensity or colour values of both p and q. For similar neighbouring pixels, this value should be smaller in comparison to the dissimilar neighbouring pixels. Spatial smoothness term is chosen as in:

$$\mathcal{D}_{p,q} = \sum_{\{p,q\in\mathcal{N}\}} e^{\frac{-||I(p)-I(q)||^2}{2\sigma^2}} * \delta_{p,q} \tag{3.11}$$

and,

$$\delta_{p,q} = \begin{cases} 1 & if\ x_p \neq x_q \\ 0 & else \end{cases}$$

where, I(p) and I(q) are intensity values of pixel p and q and σ is variance [57]. Using Potts model (δ) discontinuity between any pair of labels is penalised equally.



(a) image

(b) segmentation without pairwise potential

(c) segmentation with pairwise potential

(d) image with Gaussian noise ($\mu = 0, \sigma = 0.01$)

(e) segmentation without pairwise potential

(f) segmentation with pairwise potential

Figure 3.8: Influence of the pairwise potential on segmentation.

Figure 3.8 illustrates the influence of pairwise potential on segmentation results. It is clear from the figure that *segments* boundaries are smooth with pairwise potentials. Furthermore, segmentation results improved significantly for the noisy images.

3.4.3 Higher order clique potential

A higher order clique potential is an energy term created using three or more neighbouring pixels. This potential favours all pixels belonging to a clique taking the same label. Methods for assigning labels to a clique can be categorised into two groups. The first category of approaches are influenced by a *hard* constraint that assumes that all pixels of a particular clique belong to the same physical object and therefore a single label should be assigned to them [98, 99, 100]. This is not always true, especially near the object boundaries where cliques may have pixels belonging to the different objects. In the second category of approaches, a *soft* constraint allows label propagation methods to assign different labels to the pixels belonging

to the clique [101, 102, 58, 103, 1].

In general, label propagation methods [104, 58, 49, 99, 1] create higher order cliques using unsupervised segmentation algorithms, e.g., the *Pb* edge detector [34] and the mean shift algorithm [105]. To enforce the soft constraint, higher order cliques are created by combining multiple segmentation results of the same image. These multiple segmentations can be created either by multiple unsupervised segmentation methods or by a single unsupervised segmentation method with different parameter settings. In another type of approach, Kohli et al. [17] defined higher order cliques using 5×5 overlapping blocks of pixels of the input image.

We have created the higher order cliques by dividing the image into overlapping windows of $n \times n$ pixels as method proposed by Kohli et al. [17]. Their method is selected because cliques created using the inner pixels of the *segments* can easily be removed from the minimisation process using the spatio-temporal displacement constraint (Equation 3.14). Thus, reducing the overall computation cost. Pixels of any clique can belong to different objects therefore, higher order clique potentials $D_c(x_c)$ are implemented as *soft* constraint similar to the robust $\mathcal{P}^n$ Potts model as:

$$\mathcal{D}_c(x_c) = \begin{cases} N_i(x_c)\frac{1}{Q}\gamma_{max} & if N_i(x_c) \leq Q \\ \gamma_{max} & otherwise \end{cases} \tag{3.12}$$

where, $N_i(x_c)$ denotes the number of variables in the clique c not taking the dominant label, i.e., $N_i(x_c) = min_k(|c| - n_k(x_c))$ and $\gamma_{max} = |c|^{\theta_\alpha}(\theta_p^h + \theta_v^h G(c))$; $Q = .1 \times c$ is the truncation parameter which controls the rigidity of the higher order clique potential. $Q = .1 \times c$ means only 10% of pixels can have different labels in the clique c; $|c|$ is cardinality of clique c; $n_k(x_c)$ denotes the number of pixels taking dominant label k in c; $G(c)$ is a measure to the quality of clique and; θ_α, θ_p^h and θ_v^h are weighting parameters [58]. The quality of a clique $G(c)$ is measured as a response of unary feature (in our case colour and texture) of all constituent pixels of a clique c as:

$$G(c) = exp(-\theta_\beta^h \frac{||\sum_{i \in c}(f(i) - \mu)^2||}{|c|}) \tag{3.13}$$

where $\mu = \frac{\sum_{i \in c} f(i)}{|c|}$, $f()$ is a function evaluated on all constituent pixels if the clique c and θ_β^h is a weighting parameter.

Under the spatio temporal displacement constraint (Section 3.3), for each clique c label propagation should favour common labels which are present in the label subset of each pixel of c. It should not assign a label to pixel p of c that is not present in the label subset $\mathcal{L}^p$ of p. To enforce this spatio temporal displacement constraint, the higher order clique potential is computed by multiplying the robust $\mathcal{P}^n$ Potts model (Equation 3.12) with the a function δ as:

$$\mathcal{D}_c(x_c) = \delta_{\{c,x_c\}} * \mathcal{D}_c(x_c) \tag{3.14}$$

and

$$\delta_{\{c,x_c\}} = \begin{cases} 1 & if\ x_p \in \mathcal{L}^p_{t-1}\ \forall p \in c \\ \infty & otherwise \end{cases}$$

For each iteration of the α-expansion in our algorithm, a higher order clique is only created if more than one label, including the label α, is present in spatio temporal neighbourhood of all constituting pixels. Therefore, no higher order clique is added in the region where only one label can be assigned to the pixels. This condition reduces the number of higher order cliques and hence reduces the computation expense. However, the computation cost of the label propagation methods with higher order clique, unary and pairwise potentials will be more than the computation cost of the label propagation methods with only the unary and the pairwise potentials.

3.5 Results

Our label propagation starts with the learning of colour (GMM) and texture (normalised texton histogram) models for each *segment* present in the manually specified segmentation of the first processed frame. Figures 3.9 and 3.11 are examples of the manually specified segmentation images created using the first processed frame from the *flower and garden* and the *boy* sequences, respectively. We processed 50 frames of the *flower and garden* and 191 frames of the *boy* video sequences. In all our label propagation experiments we use following parameter setting: $d_{max} = 10$, $\theta_{col} = 0.5$, $\theta_{tex} = 0.5$ $\theta_\alpha = 0.8$, $\theta^h_p = 0.4$ and $\theta^h_v = 0.5$. A cross-validation method is used for selecting parameters values. Parameters values are chosen from the range given in the table 3.1 and values giving the best segmentation results are

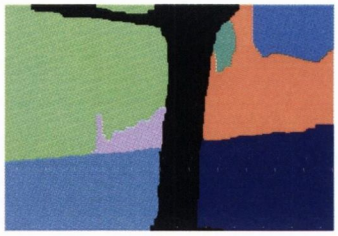
selected.

No.	Equation	Variable	Range	Purpose
1.	3.7	θ_{col}	$[0, 1]$	weight to set priority of the colour cues in unary energy potential.
2.	3.7	θ_{tex}	$[0, 1]$ & $1 - \theta_{col}$	weight to set priority of the texture cues in unary energy potential.
3.	3.12	c	[3, number of pixels in a video frame]	variable to represents the size of higher of clique. Size must be greater than 2. In the case of less than 2 pixels, higher order clique is same as a pixel pair.
4.	3.12	Q	$n \times c$ i.e. $0 < n \leq 1$	variable to represents the number of pixels in higher order clique which can have labels different from the dominant label. This variable controls rigidity of the clique.
5.	3.12	θ_v^h	$[0, 1]$	weight to set the importance of clique quality function $G(c)$ in the higher order clique potential.
6.	3.12	θ_p^h	$[0, \infty]$	variable to adjust the clique quality function $G(c)$ according to the values of unary and pairwise potentials. A high value means giving more importance to the higher order clique potentials in comparison of the unary and the pairwise potentials during the label propagation.
7.	3.12	θ_α^h	$[0, 1]$	variable to adjust the clique quality function $G(c)$ according to the values of unary and pariwise potentials. A high value means giving more importance to the higher order clique potentials in comparison of the unary and the pairwise potentials during the label propagation.
8.	3.13	θ_β^h	$[0, \infty]$	variable to set weight to the variance sensitive function composed of unary cost.

Table 3.1: User defined variables, their ranges and purposes.

We have minimized energy function (Equation 3.1) using α-expansion [16, 17] as discussed in the previous section to obtain labelling of other frames of the sequence. C++ and MATLAB® implementation provided by Kohli et. al [58] is used for this purpose. Figures 3.10 and 3.12 show five frames of *flower and garden*; and *boy* video sequences with label propagation results of α-expansion method proposed by Boykov et al. [16], Wang and Collomosse's method [1] and our method in first, second, third and fourth rows respectively. Row five of figure 3.12 shows our label propagation results with handling appearance of new objects. In each figure, segmentation results shown in row 3 are taken from the video material published in [1] with the authors permission. Both α-expansion and our segmentation methods start with the manual segmentation of the first processed frame of the sequence.

(a) First processed frame



(b) Manually specified segmentation

Figure 3.9: The first processed frame of the *flower and garden* sequence and the corresponding manually specified segmentation. In the manually specified segmentation, the first frame is divided into eight *segments* and each *segment* is highlighted in different colour.

(a) Frames

(b) Label propagation results using the α-expansion method [16].

(c) Label propagation results using Wang and Collomosse's method [1].

(d) Our label propagation results.

(e) Our label propagation results with handling appearance of new objects. Methods for handling appearance of new objects are detailed in section 3.6.

Figure 3.10: Label propagation results for five frames of the *flower and garden* sequence. Frames are selected at regular interval of 5 frames.

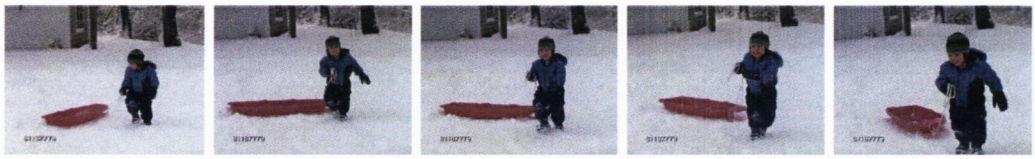
The first processed frame of the *flower and garden* video is divided into eight *segments* (Figure 3.9). α-expansion (Figure 3.10b) and our (Figure 3.10d) segmentation method use this ground truth for learning properties, e.g., colour and texture, of *segments* (learned label set). It is observed that two of these *segments*; the flower bed to the left hand side of tree and the flower bed to the right hand side of the tree; have similar colour and texture properties. In α-expansion algorithm based methods, each label present in the learned label set should have unique properties (colour and texture in our case). If there are two *segments* with similar properties, then these algorithms will assign the label of one of the *segments* to the pixels belonging to both *segments*. Therefore, a single label is assigned to both the *segments* representing the flower bed in next frames. However, with the help of the spatio temporal displacement constraint our label propagation algorithm is able to propagate the labels present in the learned label set including labels of the *similar* segments.

In the video sequence, a portion of the sky started appearing in between the top right branches of the tree. Our label prorogation method misclassified these pixels (row 4). Label of *segment* representing the tree is assigned to these pixels because only label of the tree is present in their spatio-temporal neighbourhoods. Last row shows results after processing the frame with our methods for handling the appearance of new objects in the video sequence. A new label is assigned to the misclassified pixels belonging to the sky. Our methods for handling appearance of new objects are detailed in section 3.6. However, if the colour and the texture properties of pixels belonging to two neighbouring *segments* are similar, our label propagation method will misclassify some or all pixels from these *segments*.

The first processed frame of the *boy* video sequence is divided into eleven *segments* (Figure 3.11). α-expansion (Figure 3.12b) and our (Figure 3.12d) segmentation methods start label propagation with the learned label set of eleven labels. In Wang and Collomosse's results, similar colours are used to represent different *segments* making it hard to see the *segments* clearly (Figure 3.12d). Therefore, we have used different colours to represent the *segments* in both α-expansion [16] results and our results.

(a) First processed frame (b) Manually specified segmentation

Figure 3.11: The first processed frames of the *boy* sequence in the forward direction and the corresponding manually specified segmentation. In the manually specified segmentation, the first is divided into twelve *segments* and each *segment* is highlighted in different colour.



(a) Frames

(b) Label propagation results using the α-expansion method [16].

(c) Label propagation results using Wang and Collomosse's method [1].

(d) Our label propagation results

Figure 3.12: Label propagation results for five frames of the *boy* video sequence. Frames are selected at regular interval of 40 frames.

It is observed from figures that our developed method and Wang and Collomosse's method, performed better in terms of accuracy and temporal consistency of the segmentation results when compared to α-expansion method. Our results are comparable with Wang and Collomosse's results even though we do not use motion information and our higher order clique

are created using overlapping windows rather than unsupervised segmentation methods. Objective analysis detailed in chapter 7 also supports this subjective analysis. In chapter 7, our final label propagation results are compared with the α-expansion results in terms of temporal consistency. Evaluation metrics require segmentation results in form of label masks. Wang and Collomosse's results are not available in the required format (available as video) therefore, cannot be used for the objective analysis.

3.6 Incorporating new objects in label propagation

Label propagation based methods create a label set $\mathcal{L}$ by learning properties of the objects in advance using training either on test videos or on ground truth data of a few (1 or 2) previous frames. During label propagation, pixels of the current frame can only get labels present in $\mathcal{L}$. In a video sequence, new objects (which were not present in the previous frame) may appear over time. A new object appeared in the current frame could be an instance of either a learned label class or an unknown class (not present in training data). *Physical class based* label propagation can assign any label from the learned label set to any pixel of the current frame. Therefore, these methods can handle appearance of new instance of any learned label class in the current frame. In our approach (section 3.3), each pixel of the current frame is compared only with a small number of labels which are present in its spatio-temporal neighbourhood. Therefore, incorrect label may be assigned to pixels belonging to the new object if label of the class is not in present in the spatio-temporal neighbourhood of pixels. In the second case both physical class based and our method will assign incorrect labels to the pixels belonging to the new objects because each pixel p of I_t can get label only from the existing set of labels $\mathcal{L}$.

Figure 3.13 shows an example of label propagation results of our method (Section 3.1) using a video sequence from PETS 2001 data set [106] . Our label propagation method learns the properties, i.e., colour and texture, of the objects present in the video using the manually annotated ground truth data of the first processed frame. Ground truth data has divided the first frame into seven segments. As a result our label propagation starts with seven labels in label set $\mathcal{L}$. During the label propagation labels from $\mathcal{L}$ is assigned to each pixel of the current frame. In second frame, a new object (car) starts to appearing from the right side (Figure 3.13a). Our label propagation can only assign labels from $\mathcal{L}$ therefore, label representing the

road is assigned to the pixels belonging to the car (Figures 3.13b and 3.13c).

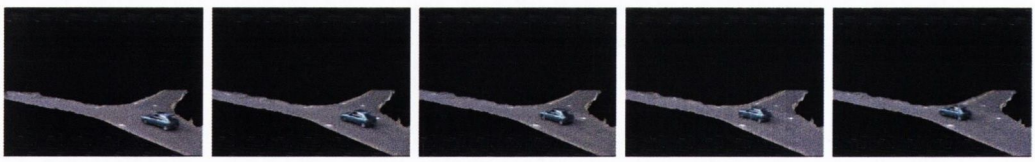
(a) Frames 1 – 5

(b) Label propagation results using our method proposed in section 3.1.

(c) *Image masks* of the *segment* representing the road and the car in segmentation results.

(d) Frames 6 – 10

(e) Label propagation results using our method proposed in section 3.1.



(f) *Image masks* of the *segment* representing the road and the car in segmentation results.

Figure 3.13: Label propagation method presented in section 3.1 can not handle appearance of new objects. A new object, car starts to appear in the second frame. Our label propagation method assigns label of the *segment* representing the road to the pixels belonging to the car.

In this section, we present a novel approach to deal with new objects in label propagation. Our approach has following steps:

Algorithm 1 Handling appearance of new objects

1: Add a new label called *void* label into the label set $\mathcal{L}' = \{\mathcal{L}, void\}$ (Section 3.6.1).

2: Update data term for existing labels $\forall \alpha \in \mathcal{L}$ (Section 3.6.2).

3: Detect pixels belonging to new objects (Section 3.6.3).

4: Create labels for new objects and update the label set $\mathcal{L} = \{\mathcal{L}, newLabels\}$ (Section 3.6.4).

5: Rerun the label propagation from I_{t-1} to I_t with updated label set $\mathcal{L}$ (Section 3.6.5).

Algorithm 1 is a pre-processing step for handling the appearance of new object over time. This step adds an additional computation cost to our label propagation method. To reduce the computation cost of this step only colour cues (texture cues ignored) are used as unary potential. New object is detected only when the colour properties of pixels belonging to the new objects are statistically different from the colour properties of learned label set.

3.6.1 Void label

In label propagation, to deal with the new objects, we add a new label called *void* label in the label subset $\mathcal{L}_{t-1}^{p}$ of all pixels $p \in \mathcal{P}$. Pixels belonging to new objects should be assigned *void* label in label propagation. Data cost for assigning void label, $x_p = void$ to any pixel p can be given as:

$$\mathcal{D}_p(x_p) = \gamma \tag{3.15}$$

where, γ non negative.

3.6.2 Updating existing data term

The data term of the energy equation encourages pixels to be assigned labels of similar coloured objects. We have used Gaussian Mixture Model (GMM) to capture the colour distributions of each label where each mode of GMM is a two dimensional (one for each colour channel) Gaussian distribution. In statistics, the *three sigma* rule states that for a normal distribution about $68.27\%, 95.45\%$ and 99.73% of all values lies within 1, 2, and 3 standard deviation of the mean, respectively. Using three sigma rule we argue that, if a pixel belongs to a label, the pixel's colour should lie within $n = 1$ *or* $2or$ 3 standard deviations away from the mean of *atleast* one mode of the label's colour GMM. Otherwise, pixel does not belong to the label. We have used *Mahalanbois distance* to measure the distance of the

pixel's colour value from the mean. Based on this argument, we updated the data term of existing labels as follows:

$$\mathcal{D}_p(x_p) = \delta_p(x_p) * \mathcal{D}_p(x_p) \tag{3.16}$$

where,

$$\delta_p(x_p) = \begin{cases} 1 & if\ \exists(\mu_{xk}, \Sigma_{xk}) : \sqrt{(I(p) - \mu_{xk})^T \Sigma_{xk}^{-1} (I(p) - \mu_{xk})} \leq n\ \forall k \\ \infty & else \end{cases}$$

where, n equals to 1 *or* 2*or* 3; mean and covariance parameters of k^{th} component of GMM of label x_p are represented by μ_{xk} and Σ_{xk}. Equation 3.16 enforces the condition that a label can't be assigned to a pixel unless its colour value is within n standard deviations from atleast one mode of label's colour GMM.

3.6.3 Detecting pixels belonging to new objects

In the previous section, we argued that a pixel p can only belong to a new object if colour of p is not within n standard deviations from atleast one mode of colour GMM of any label present in its spatio temporal neighbourhood. Therefore, label propagation from I_{t-1} to I_t with the updated label set $\mathcal{L}' = \{\mathcal{L}, void\}$ should mark all pixels belonging to new objects as *void*.

In figure 3.14, to construct the *st* graph for label $x_p = \alpha$, a t-link between pixel p and source terminal will only be created if $\mathcal{D}_p(x_p) \neq \infty$. In label propagation, a pixel p will not connect to any source terminals until $s = void$ because for all other labels value of data cost is ∞, i.e., $\mathcal{D}_p(\alpha) = \infty$, $\forall \alpha \in \mathcal{L}$. In case of $s = void$, p will connect only the source because $\delta_p(x_p)$ is not ∞. Hence, in label propagation only *void* label can be assigned to the pixel p.

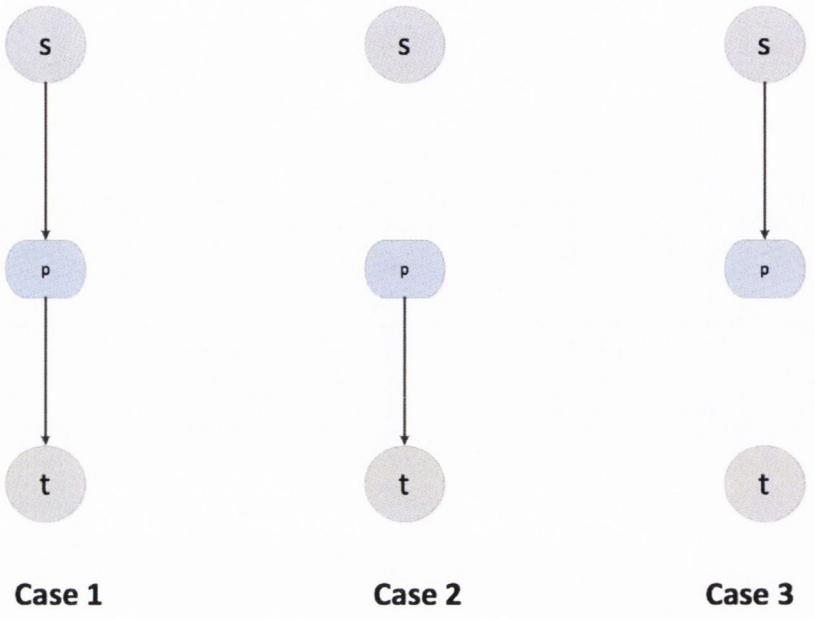


Figure 3.14: *st* graph showing only one pixel p for label propagation. (a) For a normal pixel p: $\mathcal{D}_p(x_p) \neq \infty$ so p is connected with both source s and sink t terminals and label of source or sink terminal can be assigned to p. (b) For pixel p belonging to the new object: $\mathcal{D}_p(x_p) = \infty$, $\forall \alpha \in \mathcal{L}$ and when label of source $s \neq void$, p is not connected to source terminal. Label of s cannot be assigned to p; (c) source $s = void$ so p connected to source so label of source $s = void$ will be assigned to p.

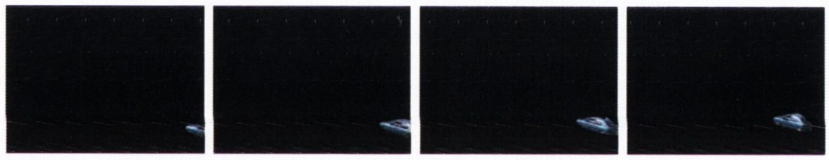
3.6.4 Creating labels for new objects

The label propagation algorithm will mark all the pixels belonging to the new objects with the *void* label. These pixels may belong to different new objects. Therefore, to deal with appearance of multiple new objects (*voids*) at the same time, the Mean-Shift algorithm in space-colour domain is used for clustering the marked pixels. To avoid segmentation errors, only clusters bigger than a *minimum area* are treated as new objects. Value of this *minimum area* depends on average size of the *segments*. For clusters bigger than the *minimum area*, our algorithm learns colour and texture cues and creates new labels in the label set. These new labels are also added in the label subset $\mathcal{L}^p_{t-1}$ of each pixel p of the respective clusters.

(a) Frames 1 – 5

(b) Label propagation results

(c) *Image masks* of the *segment* representing the road in segmentation results.



(d) *Image masks* of the *segment* representing the car in segmentation results.

(e) Frames 6 – 10

(f) Label propagation results

(g) *Image masks* of the *segment* representing the road in segmentation results.

(h) *Image masks* of the *segment* representing the car in segmentation results.

Figure 3.13: Label propagation with the ability of handling the appearance of new objects during the video sequence. With the approach developed in the current section, our label propagation algorithm is able to detect the appearance of the car and consequently assigned a new label to the pixels belonging to the car. This is a clear improvement over the segmentation results shown in figure 3.13. In this example, frame size is 240×352. Our algorithm assigned a new label to cluster when its area is bigger than 50 pixels.

3.6.5 Rerunning the label propagation with updated label set

After creating new labels for new objects and learning their properties, i.e., colour and texture, our algorithm rerun the label propagation from I_{t-1} to I_t with updated label set. In the label propagation, the colour, the texture and the brightness properties of all *segments* are used for assigning the best labelling to the video frames. New labels are already detected in the previous iteration of label propagation hence the updated label set does not have *void* label. In this iteration of label propagation, the data term is computed as mentioned in section (3.4.2). Figure 3.13, shows the results after the label propagation is run following label set update. Our algorithm successfully handled appearance of the car in the second frame. A new label is assigned to pixels belonging to the car (Figures 3.15d and 3.14h). This is a clear improvement over the segmentation results shown in figure 3.13.

3.7 Conclusion

In this chapter we have described one of our major contributions, spatio temporal displacement constraint and handling new objects during label propagation. Our algorithm learned the colour and the texture properties of the *segments* using the manual segmentation of the first processed frame. The best labelling was achieved by minimising an energy equation with unary, pairwise and higher order clique potentials using α-expansion for each frame of the video. Segmentation results achieved by our method are temporally consistent and comparable with two state of the art methods: α-expansion method [16] and the method proposed by Wang and Collomosse [1], even though motion cue is not used during the label propagation.

In a video sequence, new objects can appear over time. New objects can be an instance

of a learned label class or a unknown class (not present in training data). α-expansion [16] can assign any label from the learned label set to any pixel of the current frame. Therefore, it can handle appearance of new instance of any learned label class in the current frame. However, α-expansion will also assign the same label to all instance of a physical object even though unique labels are assigned to different instances of the physical object in the manually specified segmentation of the first processed frame. If new object belongs to a unknown class, the α-expansion [16] will assign incorrect labels to the pixels belonging to the new objects. In our approach, each pixel of the current frame is compared only with a small number of labels which are present in its spatio-temporal neighbourhood. As a result, developed method is able to assign unique labels to all instances of a physical object. However, a pre-processing step is required to handle appearance of new objects of both known and unknown classes. This pre-processing step adds additional computational cost in our approach.

Our algorithm used only the brightness, the colour and the texture cues of the *segments* during the label propagation. It performed substantially better when the colour and the texture cues of pixels belonging to the neighbouring *segments* were different. Otherwise, it may misclassify some of the pixels belonging to neighbouring *segments*. This situation can be handled by using more cues, for example shape and motion, during the label propagation. In the next chapter, we present an algorithm first to learn such cues specifically shape and motion, and another to use these cues for improving the label propagation results for rigid objects.

Chapter 4

Improving label propagation results for rigid objects

In the previous chapter, our label propagation method uses brightness, colour and texture cues to assign labels to each pixel of the current frame. Normally, a single label is assigned to spatially connected homogeneous pixels. A group of such pixels is called a *segment*. A *segment* should represent a complete physical object or a part of a physical object but it should not have pixels belonging to more than one physical object. If the colour and texture of pixels belonging to two neighbouring *segments* are similar, label propagation methods such as α-expansion [16], Wang et al. [1] and our methods on the previous chapter may misclassify some or all of the pixels from these *segments*. There are two cases when this can happen. In the first case, different parts of the same object which were separated by other (foreground) objects until time $t-1$ are now spatially connected in the current frame due to the relative changes in the positions of the object and other (foreground) objects. In this case the *segments* representing spatially disconnected parts of the object should be merged to create one *segment* (Section 4.4). In the second case, *segments* with similar colour and texture properties which become (or are) neighbours should not be merged if they represent different physical objects (Section 4.2).

Figures 4.-1 illustrates a limitation of our label propagation method which was presented in chapter 3 using ten frames of the *car1* video sequence of the Berkeley motion segmentation dataset [107]. In this video sequence, a moving car is captured by a moving camera. It can be seen in the figure that both the colour and texture properties of several pixels near the front of the car and the road are similar. Therefore, during label propagation our method can

assign either label to such pixels. The label of the *segment* representing the road is assigned to a few pixels belonging to front of the car and the label of the *segment* representing the car is assigned to a few pixels belonging to the road at the back of the car (Figures 4.1c, 4.1d). However, pixels belonging to the *segment* representing the trees in background are labelled correctly because their colour and texture properties are very different from the remaining two *segments* (Figure 4.1e).

The initial label propagation results achieved using our label propagation method may be incorrect but these problems can be addressed using new cues like the motion and the shape of the *segments*. These cues can be used to improve initial label propagation results and therefore, can help to improve the accuracy and the temporal consistency of the segmentation results. In this chapter, we present algorithms to improve label propagation results for rigid objects using motion and shape cues. Our motion and shape cues rely equally on both the spatial and the temporal information present in video. We assume that all the pixels of each *segment* have same rigid 2D apparent motion. Our developed method will not work for non rigid motion and multiple motions within a *segment*.

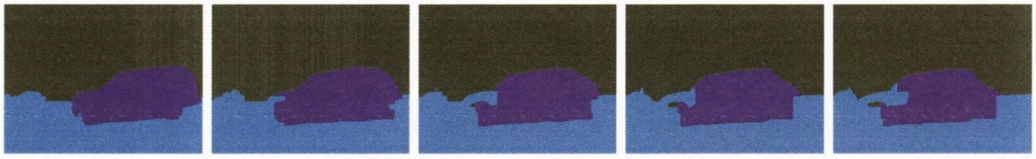
Chapter outline: Section 4.1 present a review of methods that can be used for learning motion cues of the *segment*. Our algorithm for improving the label propagation results is presented in section 4.2. Once the label propagation results are fixed, a method for learning shape cues is presented in section 4.3. Section 4.4, presents a post processing step to merge the *segments* using shape, colour, texture and motion features and finally results are presented in section 4.5.

4.1 Learning motion of *segments* between consecutive frames

Apparent motion information of *segments* or pixels or feature points provides cue for the temporal direction. Using motion, label propagation methods can predict the locations of the *segments* in the next frame. This predicted locational information has been used along with other spatial cues (i.e., colour and texture) to improve the temporal consistency of the segmentation results [49, 1]. To improve our label propagation results for the current frame we learn motion information for each *segment* present in the current frame. In a video sequence, shape of the *segments* may change due to both motion (e.g., rotation, scaling) and non motion (e.g., lens distortion) effects. A lens distortion may deform the *segments* by sketching

or shrinking their boundaries. Effects of lens distortion can be removed satisfactorily from video frames if intrinsic and extrinsic parameters of the camera are known.

(a) Frames 1 – 5



(b) Label propagation results using our method proposed in chapter 3.

(c) *Image masks* for the *segment* representing the car in the segmentation result of the first frame.

(d) *Image masks* for the *segment* representing the road in the segmentation result of the first frame.

(e) *Image masks* for the *segment* representing the trees in the segmentation result of the first frame.

(f) Frames 6 – 10

(g) Label propagation results using our method proposed in chapter 3.

(h) *Image masks* for the *segment* representing the car in the segmentation result of the first frame.

(i) *Image masks* for the *segment* representing the road in the segmentation result of the first frame.

(j) *Image masks* for the *segment* representing the trees in the segmentation result of the first frame.

Figure 4.-1: Example of inaccurate *segments* merging (partially). Ten frames of the *car1* video sequence of Berkeley motion segmentation dataset [107] are used in this example. Our label propagation algorithm has divided each frame in 3 *segments*; (c) - (e) and (h) - (j) show *image masks* of each of the *segment*. The *image mask* of a *segment* has the colour values of the pixels belonging to the *segment* in a particular frame.

For the approached described in this thesis, any method (optical flow covered in section 4.1.1 or geometrical transformation covered in section 4.1.2 or frequency transform covered in section 4.1.3) can be used to determine apparent motion information of the *segments*. We have used *imregister*[1] function of MATLAB® to estimate $2D$ affine motion parameters for all the *segments* of the current frame.

4.1.1 Optical flow

In the simplest form, motion information of pixels present in the frames of a video can be described by optical flow. In 2D space, optical flow of any pixel or point represents the projected displacement $(\vec{u}, \vec{v})$ of that pixel or point between two frames where $\vec{u}$ and $\vec{v}$ represent the displacements in the horizontal and vertical axes respectively. In reality, the vector field $(\vec{u}, \vec{v})$ does not always correspond to real motion of pixels or points, as it may be affected by instability in illumination [109].

[1] *imregister* function is part of the Image Processing Toolbox of MATLAB® and its implementation is based on the work of Styner et al. [108].

4.1.2 Geometric transformation

This approach to motion estimation uses the underlying geometrical deformation of a *segment* between two frames to estimate its motion. Translation is the simplest geometrical deformation and in the translation motion model, *segments* move without any change in their orientations with respect to a fixed point. Affine (six parameters) and perspective (eight parameters) models tackle higher order transformations. We have used affine transformation to model motion information. The standard affine transformation model in $2D$ space uses six parameters, corresponding to scaling, rotation, shear and translation to model transformation of a *segment* between frames at time t and $t+1$:

$$T\begin{bmatrix} x \\ y \end{bmatrix} = A\begin{bmatrix} x \\ y \end{bmatrix} + u \tag{4.1}$$

where A and u represent the linear part and the translational part of the transformation, respectively.

Parameter estimation for an affine transformation is an optimisation problem. The optimisation process starts with an initial set of parameters. Similarity metric results of the previous iteration are used to estimate a new set of parameters in such a way that this new set of parameters improves the similarity score. The metric used to measure similarity is called the *similarity metric*. Euclidean distance [110] or correlation [108] or normalised cross correlation [11, 111] between the *segment* and the transformed *segment* in feature space can be used as the similarity metric.

Optimisation of the similarity metric is guided either by considering all pixels or just by considering *feature points* of the *segment*. Pixel by pixel comparison is time/resource consuming but it can provide a good approximation of the transformation parameters for both textured or non textured *segments*. One way to reduce the computation time is to use a hierarchical model where, pixel by pixel comparisons at lower resolutions guide the optimisation process at the higher resolutions. Feature point based methods reduce the number of points for comparisons by using small numbers of unique features like, corners, boundary points [110, 112], points [113, 114] and edges [115]. In practice, these methods can only produce robust transformation parameters for feature rich *segments*.

4.1.3 Frequency domain

Other approaches to motion parameter estimation methods reduce the problem of estimating $2D$ transformation parameters to the estimation of $1D$ transformation parameters by using properties of the *segments* in frequency domain. These methods use transforms, such as the Fourier transform [116, 117], Hough transform, Radon transform [118] and Trace transform [119] to transform the *segments*. For example, in the Fourier domain simple translation parameters (2 parameters) are estimated regardless of translation, rotation, scaling or shear changes.

4.2 Improving label propagation results

Our label propagation method presented in chapter 3 cannot handle situations where neighbouring *segments* have pixels with similar colour and texture properties near their shared boundaries, and as a result, incorrect labels may be assigned to such pixels. In this section, we present a novel method to improve these label propagation results. Our method uses the shape and the motion information of the *segments* to fix the label propagation errors. In the first step, a *mask*, $\theta_{\{t-1,t\}}$ is created at time t by applying the determined motion to the previous *label mask* of each *segment*. The *label mask* of any *segment* represents shape information of the *segment* at any particular time t. $\theta_{\{t-1,t\}}$ supports multiple labels at any pixel position. Multiple labels at any pixel position indicate that current position is claimed by more than one *segment* and objective of our method is to find out the correct labels for such positions. For example, pixels belonging to an occluded region will have labels of the foreground and the background *segments*. In the second step, pixel positions with the same set of labels are grouped to create clusters. In the last step, using energy minimisation our algorithm assigns a label to each pixel of each cluster. The final label masks support only a single label for each pixel position. Our three stage algorithm is:

Algorithm 2 Refine labelling

1: Create a label mask, $\theta_{\{t-1,t\}}$ supporting multiple labels at each pixel position using segmentation results of the previous and the current frames (Section 4.2.1).

2: Cluster pixels of $\theta_{\{t-1,t\}}$ having the same set of labels together to create a cluster map $\mathcal{C}$ i.e., each cluster centre represents a unique set of labels $\mathcal{L}^c$ (Section 4.2.2)

3: Using energy minimisation, assign labels to the pixels of each cluster of $\mathcal{C}$ i.e $l \in c_l$ (Section 4.2.3)

4.2.1 Label mask supporting multiple labels

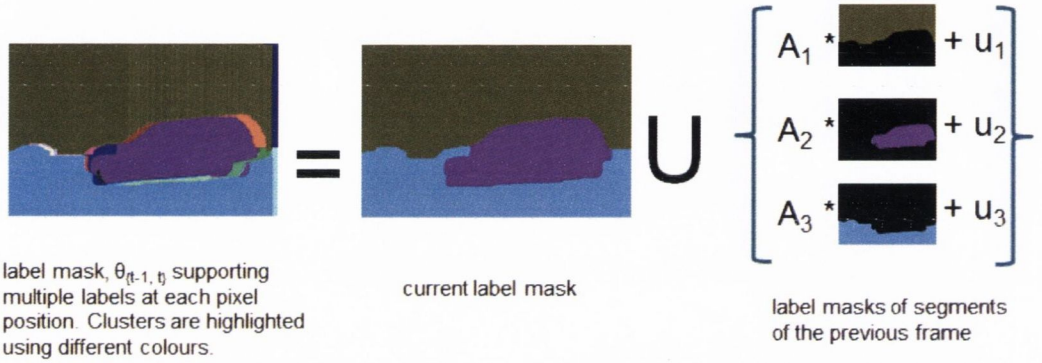


Figure 4.0: Creation of a label mask supporting multiple labels using the current label mask, label masks of the previous frames and motion parameters.

After learning the motion of each *segment* in section 4.1 using label masks of the *segments* present in the current frame and the previous frame, algorithm 3 applies motion to the *label mask* of each *segment* of the previous frame to create warped label masks, $M_{\{t-1,t\}}$. Further a *mask*, $\theta_{\{t-1,t\}}$ is created using these warped label masks where $\theta_{\{t-1,t\}}$ supports multiple labels for each pixel position. It has more than one labels for the pixel positions where the warped label masks of more than one *segment* overlap. If $\theta_{\{t-1,t\}}$ is created only using the segmentation results of the previous frame, information of deoccluded parts of the *segments* which are visible for the first time in the current frame I_t would be missing from it. To deal with this issue algorithm updates $\theta_{\{t-1,t\}}$ using the *label mask* M_t of the current frame as determined in chapter 3. For any pixel position p, the label present at p in M_t is added to $\theta_{\{t-1,t\}}$ if $M_t(p) \notin \theta_{\{t-1,t\}}(p)$. Figure 4.0, shows creation of $\theta_{\{t-1,t\}}$ using the current label mask, label masks of the previous frames and motion parameters.

Algorithm 3 Create a label mask supporting multiple labels at each pixel position

1: **for** each *segment* l of the current frame **do**

2: get the *label mask*, M^l_{t-1} of l from the segmentation results of the previous frame. The label mask of a *segment* only has shape information of the *segment* present in a particular frame

3: determine affine motion model parameters $(A^l_{\{t-1,t\}}, u^l_{\{t-1,t\}})$ for *segment* l between the previous and the current frames (Section 4.1)

4: create warped label mask, $M^l_{\{t-1,t\}}$ by applying motion to M^l_{t-1}:

$$M^l_{\{t-1,t\}} = A^l_{\{t-1,t\}} * M^l_{t-1} + u^l_{\{t-1,t\}} \tag{4.2}$$

5: update label mask, $\theta_{\{t-1,t\}}$. $\theta_{\{t-1,t\}}$ supports multiple labels at each pixel position:

$$\theta_{\{t-1,t\}}(p) = \{\theta_{\{t-1,t\}}(p),\ M^l_{\{t-1,t\}}(p)\} \tag{4.3}$$

6: **end for**

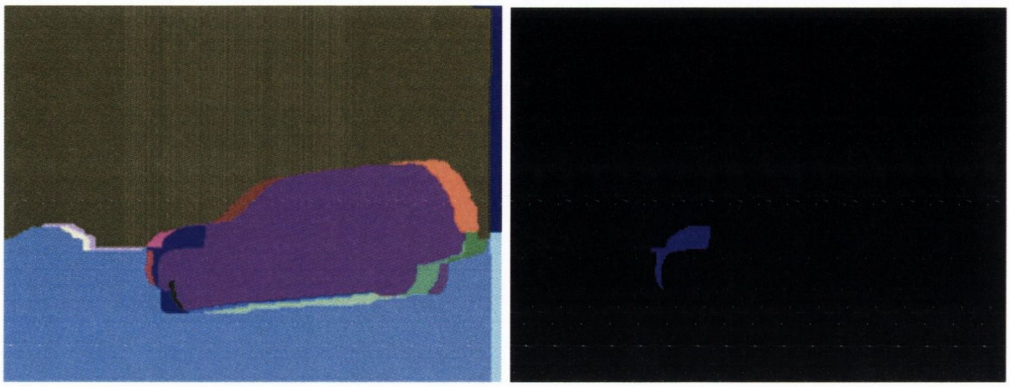
7: update $\theta_{\{t-1,t\}}$ to incorporate deocclusion using the segmentation result M_t at time t:

$$\theta_{\{t-1,t\}}(x) = \begin{cases} \{\theta_{\{t-1,t\}}(p),\ M_t(p)\} & if\ M_t(p) \notin \theta_{\{t-1,t\}}(p) \\ do\ nothing & otherwise \end{cases}$$

4.2.2 Creation of clusters

The *mask* $\theta_{\{t-1,t\}}$ created in the previous section supports multiple labels at each pixel position and our final goal is to assign a single label to each pixel of this mask. In this section, a cluster map $\mathcal{C}$ is created by clustering pixels of $\theta_{\{t-1,t\}}$ having the same set of labels. The set of labels, $\mathcal{L}^c$ associated with each cluster c represents possible labels that may be assigned to the pixels belonging to c. Therefore, similar to the spatio temporal displacement constraint (Section 3.3), our algorithm compares pixels of any cluster c only with the set of labels, $\mathcal{L}^c$ associated with the cluster where $\mathcal{L}^c \subset \mathcal{L}$, where $\mathcal{L}$ represents the set of learned labels. Figure 4.1, shows an example cluster from the cluster map $\mathcal{C}$ created for a frame of *car1* video sequence. The manual segmentation of the first processed frame of the *car1* video sequence has three *segments*; road, car and background (Figure 7.13). Each pixel of our example cluster supports

two labels, label of pixels belonging to the car ($l1$) and label of pixels belonging to the road ($l2$). Therefore, label subset $\mathcal{L}^c$ of the cluster has only two member $l1$ and $l2$. Label representing the road ($l3$) in the manual segmentation of the first processed frame is not a part of the $\mathcal{L}^c$. Final goal is to assign a single label from $\mathcal{L}^c$ to each pixel of the cluster.



(a) Label mask $\theta_{\{t-1,t\}}$ (b) An example cluster.

Figure 4.1: An example cluster from the cluster map $\mathcal{C}$. Two labels, label of the pixels belonging to the car and label of the pixels belonging to the road in the manual segmentation of the first processed frame are assigned to each pixel of this cluster.

Algorithms to assign a single label to all pixels belonging to any cluster using energy minimisation are detailed in the next section.

4.2.3 Assigning labels to the pixels of cluster map using energy minimisation

We define our energy function same as function defined earlier in equation 2.1:

$$E(x) = E_{data}(x) + E_{smooth}(x)$$

where, $E_{data}(x)$ measures the disagreement between labelling x and the observed data; and E_{smooth} measures the extent to which x is not piecewise smooth (spatially). We have use α-expansion [16] to minimize this energy function.

Both $E_{data}(x)$ and $E_{smooth}(x)$ can be defined by the unary and pairwise terms of the *Gibbs* energy function. Hence, E_{data} can be written as the sum of data costs of all clusters

$$E_{data}(x) = \sum_{c \in \mathcal{C}} \mathcal{D}_c(x_c) \tag{4.4}$$

where $\mathcal{D}_c(x_c)$ is a function defined on observed data that measures the cost of assigning labelling x_c to any cluster c.

In terms of the Gibbs energy, $E_{smooth}(x)$ can be written as the sum of pairwise potential functions

$$E_{smooth}(x) = \sum_{\{p,q \in \mathcal{N}\}} \mathcal{D}_{p,q}(x_p, x_q) \tag{4.5}$$

where $\mathcal{N}$ defines the neighbourhood of clusters and $\mathcal{D}_{p,q}(x_p, x_q)$ measures the cost of assigning labelling x_p, x_q to clusters p and q. In section 4.2.3.1, we present our algorithm for learning the data cost. The method for learning the pairwise potential is presented in section 4.2.3.2.

4.2.3.1 Data cost

Every cluster c in the cluster map is actually a part of an object represented by one of the *segments* of the label set $\mathcal{L}^c$. Pixels belonging to cluster c in the current frame and to one of the *segments* l, $l \subset \mathcal{L}^c$ in the previous frame represent the same part of a physical object at two different times. We define the data cost to assign any label l to a cluster c as the negative log likelihood of the normalised cross correlation between pixels belonging to cluster c in the current frame and pixels belonging to c represented by the *segment* l in the previous frame. Normalised cross correlation makes the likelihood estimation robust to lighting changes [11, 120]. Moreover, filtering the video frames using filter like diffusion filter before taking the normalised cross correlation can improve the normalized cross correlation scores by reducing the noise level.

We have developed a novel algorithm 4 to compute the data cost for each cluster of the cluster map. For any cluster c, the colour values of the pixels belonging to this cluster in the current frame are presented as $I_t(c)$. The pixels belonging to each *segment* of $\mathcal{L}^c$ in the previous frame can be determined using the respective *label masks* at time $t-1$. The position and the shape of any *segment* can change in frames due to the relative motion between the *segment* and the camera. To compare any part of the *segment* present in the current frame with the same part of the *segment* in the previous frame, both parts should be aligned. Hence, a warped image mask $I^l_{\{t-1,t\}}$ is created using the *segment's* motion parameters $\{A^l_{\{t-1,t\}}, u^l_{\{t-1,t\}}\}$ between frames I_{t-1} and I_t. Pixels belonging to cluster c in the warped image mask can be represented as $I^l_{\{t-1,t\}}(c)$. The alignment problem cannot be solved completely because the segment's affine motion parameter estimated in section 4.1 may

have minor errors. To deal with this issue a new *label mask*, $I_t(c')$ of pixels belonging to c and its neighbouring pixels (within a n pixel radius) is used to compute the data cost between cluster and any label l where $l \in \theta_{\{t-1,t\}}(x_c)$.

Any cluster c can belongs to deoccluded parts of a *segment*. Deocclusion means that a new part of the object starts appearing in the current frame. As this part of the object was not visible in the previous frame, we cannot use pixels of the previous frame to compute the data cost. In this case, our algorithm uses negative log likelihood of a constant value γ_{corr} as the data cost for c. The label of pixels belonging to any deoccluded region will only change if for any other *segment* the similarity score is higher than γ_{corr}. A high value of γ_{corr} will ensures that label of c only changes if c and any other *segment* are very similar.

Algorithm 4 Computing data cost

1: **for** each cluster c of cluster map **do**

2: **for** each label l of $\mathcal{L}^c$ **do**

3: create *image mask* I^l_{t-1} for *segment* l using M^l_{t-1} and the previous image frame I_{t-1} as:

$$I^l_{t-1}(p) = \begin{cases} I_{t-1}(p) & if\ M^l_{t-1}(p) \neq 0 \\ 0 & otherwise \end{cases} \tag{4.6}$$

4: determine affine motion model parameters $(A^l_{\{t-1,t\}}, u^l_{\{t-1,t\}})$ for *segment* l between frames I_{t-1} and I_t

5: create warped image mask, $I^l_{\{t-1,t\}}$ by applying affine motion model parameters of *segment* l to I^l_{t-1} as:

$$I^l_{\{t-1,t\}} = A^l_{\{t-1,t\}} * I^l_{t-1} + u^l_{\{t-1,t\}} \tag{4.7}$$

6: dilate cluster c to create c'

7: compute data cost:

$$\mathcal{D}_c(x_c) = \begin{cases} -log(\Gamma(I_t(c'), I^l_{\{t-1,t\}}(c))) & if\ l\ is\ not\ deoccluded\ label \\ -log(\gamma_{corr}) & otherwise \end{cases}$$

where $\Gamma()$ refers function for normalised cross correlation.

8: **end for**

9: **end for**

4.2.3.2 Pairwise potential

The spatial smoothness term encourages continuity of *segments* labelling $\mathcal{L}$ in frame I_t. However, *segment* boundaries will suffer if the label propagation method tries to enforce smoothness everywhere during label propagation. Therefore, spatial smoothness terms should be discontinuity respecting. Spatial smoothness $\mathcal{D}_{p,q}$ can be interpreted as an average discontinuity penalty between neighbouring pixels of clusters p and q [11]. The value of spatial smoothness depends on the gradients between the neighbouring pixels of p and q. For similar neighbouring pixels this value should be higher in comparison to the dissimilar neighbouring pixels. Spatial smoothness term is chosen as:

$$\mathcal{D}_{p,q} = exp(\xi\nabla(I_t(p), I_t(q))) * \delta_{p,q} \tag{4.8}$$

and,

$$\delta_{p,q} = \begin{cases} 1 & if\ x_p \neq x_q \\ 0 & else \end{cases}$$

where $\xi\nabla(I_t(p), I_t(q))$ is the average of the gradients of the neighbouring pixels of clusters p and q; $I_t(p)$ and $I_t(q)$ refers the colour values of pixels of clusters p and q in CIE Lab colour space and σ is variance [57]. Using Potts model (δ) discontinuity between any pair of labels is penalised equally.

4.3 Learning shape masks

Shape information of objects present in the video sequence can work as a cue in the segmentation process. Many video segmentation methods learn shape information through training videos and then use this in the segmentation process along with other cues like colour, texture etc [93].

Algorithm 5 Learning shape masks

1: In the first frame at time $t = 1$, initialise the *shape masks* using the *label masks* of *segments*. The *label mask* of a *segment* only has the *segment's* shape information present in a particular frame. For each label l of I_1

$$S_1^l = M_1^l \tag{4.9}$$

2: **for** t = 2 $\rightarrow$ total number of frames in the video sequence **do**

3: **for** each *segment* l of I_t **do**

4: **if** l is present in I_{t-1} **then**

5: determine motion parameters $(A_{t-1,t}^l, u_{t-1,t}^l)$ for l between frames I_{t-1} to I_t

6: create a warped shape mask, $S_{\{t-1,t\}}^l$ by applying the motion parameters to S_{t-1}^l

$$S_{\{t-1,t\}}^l = A_{\{t-1,t\}}^l * S_{t-1}^l + u_{\{t-1,t\}}^l \tag{4.10}$$

7: create shape mask of l in current frame as:

$$S_t^l = S_{\{t-1,t\}}^l \bigcup M_t^l \tag{4.11}$$

8: **end if**

9: **end for**

10: **end for**

In this section, we present a novel algorithm 5 to learn the shapes of the *segments*. We assume that all pixels of the *segments* have similar motion. At time t, the *shape mask* for each *segment* has the shape information of the *segment* in all frames from time $t = 1$ till current time. The *shape mask* of a *segment* is a $2D$ mask of the same size as the video frame, and at $t = 1$ it is initialised with the *label mask* of the *segment*. The *label mask* of a *segment* only has shape information of the *segment* present in a particular frame. The *shape masks* and the *label masks* are created with the same size as the video frame because it is easier to operate if masks and frames are of the same size. After the label propagation described in section 4.2.3, our algorithm updates the *shape masks* of the *segments* using the current *label masks*. Any new shape information revealed in the current *label masks* should be added to the *shape masks* of the respective *segments*. *Segments* can change their positions in video

frames due to relative motion between the *segments* and the camera. Therefore, both the *shape mask* and the *label mask* of the *segment* must be aligned before updating the *shape mask*. Motion parameters $\{A^l_{t-1,t}, u^l_{t-1,t}\}$ of *segment* l between frames I_{t-1} and I_t are used to create a warped shape mask, $S^l_{\{t-1,t\}}$. The *shape mask*, S^l_t of *segment* l at time t is a union of $S^l_{\{t-1,t\}}$ and the *label mask*, M^l_t of *segment* at time t. $S^l_{\{t-1,t\}}$ has the shape information of the *segment* l present in all frames from time $t = 1$ till the previous frame and M^l_t has shape information of the *segment* present in the current frame. Therefore, the *shape mask* S^l_t will have shape information (including occluded parts) of the *segment* present in all frames from time $t = 1$ till the current time.

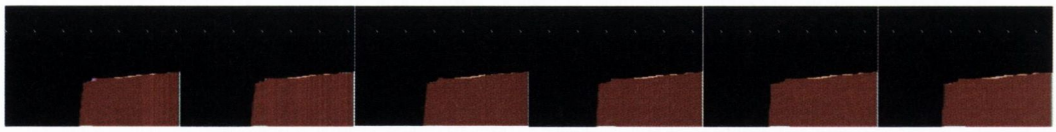
The purpose of learning the *shape masks* is to refine the labelling of video frames. The *shape masks* have a central role in the merging of *segments* (Section 4.4) and in learning both the physical explanations behind changes in the *shape masks* of *segments* and the occlusion boundaries (Chapter 5).

4.4 Region merging

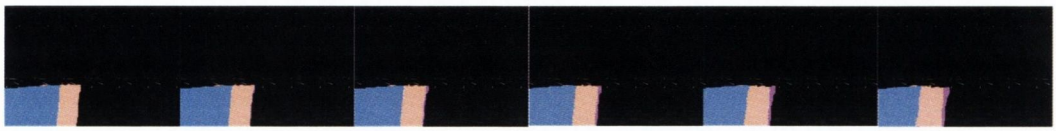
In our label propagation method, we have restricted label displacement to a maximum possible displacement (Section 3.3) depending on the average size of the *segments* and frame rate. This assumption may fail in the case of region merging. In this section, we present a post processing method to update label propagation results in case of region merging (Algorithm 6). Our method merges *segments* if their properties, i.e, colour, texture and motion, are similar and their *shape masks* are overlapping.

Overlap between *shape masks* is preferred over checking the similarity between the neighbouring *segments* because *shape masks* allow merging of *segments* which are not spatially connected in the current frame (Figure 4.2). *Shape masks* are created using frame to frame motion information of the *segments*. Motion parameter estimation is not always noise free. Error present in the motion parameters of *segments* can lead to inaccurate *shape masks*, which can further lead to false overlaps. Therefore, two measures are taken to deal with inaccuracies present in *shape masks*. Firstly, for each shape mask pixels close to boundaries (depending on area of shape mask) are removed using erosion. Secondly, merging of *segments* is only possible if the overlap area between the *shape masks* (after removing points close to their boundaries) is more than a minimum area. Value of this minimum area depends on properties such as

(a) Frames 1 – 6

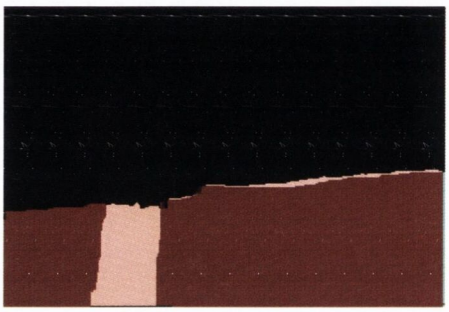


(b) *Shape masks* of the *segment* 6 (labelled as 6) for each frame.



(c) *Shape masks* of the another *segment* 3 (labelled as 3) for each frame. Velvet colour shows the overlap between the *shape masks* of the *segments* 6 and 3.

(d) Frame 7



(e) Segments 3 and 6 get merged and resulting *shape mask* of the *segment* labeled as 6.

Figure 4.2: Example of merging of two *segments* of similar properties, i.e, colour, texture and motion, using our segmentation results for the *flower and garden* video sequence. *Segments* merge when overlapping area between their *shape masks* is more than 5% the area of the smaller of *segments*.

area and relative motions of the *segments*.

In case of *segments* being merged, the label of the bigger *segment* is assigned to the pixels belonging to the smaller *segment* and the *shape mask* is updated. In coming sections 4.4.1, 4.4.2 and 4.4.3, we present the metrics used to measure similarities between the colour, texture and motion features of two *segments*.

Figure 4.2 illustrates an instance of merging two similar (based on colour, texture and motion) *segments* using our method for the *flower and garden* video sequence. The first processed frame of the *flower and garden* video is divided into eight *segments* (Figure 3.9). It is observed that two of these *segments*; the flower bed to the left hand side of tree and the flower bed to the right hand side of the tree; have similar colour and texture properties. In chapter 3, we showed that α-expansion algorithm assigned a single label to both the

segments representing the flower bed in next frames. However, with the help of the spatio-temporal displacement constraint our label propagation algorithm was able to propagate the labels present in the learned label set including labels of the *similar* segments (Figure 3.10). We called this type of segmentation *non-class* based video segmentation. Using the spatio-temporal displacement constraint our algorithm is able to handle the *non-class* based video segmentation. However, α-expansion method is not able handle the *non-class* based video segmentation.

Figures 4.2a, 4.2b and 4.2c show 6 consecutive frames of the *flower and garden* video sequence from time $t = 19 \rightarrow 24$ and the *shape masks* of two *segments*, respectively. The occluded parts are highlighted in pink in the *shape masks*. Different labels (3 and 6) are assigned to the pixels belonging to the two spatially separate segments from frame number 1 to 24. In the frame at time $t = 21$, the *shape masks* of both *segments* start overlapping (highlighted with velvet in figure 4.2c). It is also observed that relative motion of both the *segments* is same from start ($t = 1$) to the current time ($t = 21$). If two neighbouring *segments* of same properties, e.g, colour and texture, are moving together in the video than it can be referred that both the *segments* are part of a same physical object. Therefore, a single label should be assigned to both the *segment*. Our algorithm merges both *segments* when the size of the overlap area between the *shape masks* becomes bigger than 5% times the area of the smaller *segment* (earlier labelled as 3) and label of the bigger *segment* (in this case label 6) is assigned to the pixels of the smaller *segment* (Figure 4.2e). Value of minimum overlapping area is selected as 5% due to properties such as area and relative motions of the *segments*.

4.4.1 colour similarity

In our label propagation method, the colour properties of the pixels of each *segment* are modelled using the Gaussian Mixture Model (GMM) (Section 3.4.1). The two most common metrics for measuring distance between two probability density functions (PDF) are Kullback-Liebler (KL) divergence [121, 122] and the L_2 distance [110]. For two Gaussian distributions the KL divergence has a closed formed expression whereas for two GMMs no such closed form expression exists [123, 122]. A closed form expression exists for the L_2 distance between two GMMs [110] therefore we choose L_2 distance to measure the similarity between colour distributions of two *segments*. The L_2 distance between colour distributions of two *segments* p and q can be given as:

Algorithm 6 Segment merge

1: **for** each *segment* l of I_t **do**

2: get the *shape mask*, S_t^l of the *segment* l and remove points close to its boundary using erosion. S_t^l is built up from *segment's* shape information present in all frames from time t= 1 till current time

3: **for** each *segment* i of I_t except l **do**

4: get *shape mask*, S_t^i of the *segment* i

5: $overlap = true$; if number of overlapping pixels in S_t^l, after removing points close to its boundary and S_t^i are greater than a fixed percentage of the area of the smaller of *segments*

6: check *merge* conditions between *segments* l and i if:

$$mergeSegments = \begin{cases} true & if\ (overlap = similarity_{colour} = \\ & \quad similarity_{texture} = similarity_{motion} = true) \\ false & otherwise \end{cases}$$

where $similarity_{colour}$, $similarity_{texture}$ and $similarity_{motion}$ refer to the colour (Section 4.4.1), texture (Section 4.4.2) and motion (Section 4.4.3) similarities between *segments*.

7: **if** $mergeSegments == true$ **then**

8: change label of the pixels belonging to the smaller *segment* with the label of the bigger *segment* in the segmentation mask, M_t and update the *shape mask* of the bigger segment

9: **end if**

10: **end for**

11: **end for**

$$dist_{colour} = \sum (gmm(x_p) - gmm(x_q))^2 \tag{4.12}$$

where $gmm(x_p)$ and $gmm(x_q)$ refer to the Gaussian Mixture density functions constructed from the colour of pixels labelled as x_p and x_q in previous frames. colour of two *segments* can said to be similar if L_2 distance between their colour distribution is less than a constant γ_{colour}.

$$similarity_{colour} = true\ if\ dist_{colour} \leq \gamma_{colour} \tag{4.13}$$

4.4.2 Texture similarity

In section 3.4.1.2, texture information of the *segments* is gathered by applying a filter bank to each pixel of the *segments*. Texture information of any *segment* is represented as a normalised histogram of its *textons*. The chi-square significance test is used to provide a measure of the similarity between two texton histograms (h1 and h2) [95]. A null hypothesis, H_0: The two histograms are from same model, is accepted or rejected using this test. The chi-square distance between $h1$ and $h2$ is given by:

$$\chi^2 = \frac{1}{2} * \sum_{n=1}^{number\ of\ bins} \frac{(h_1(n) - h_2(n))^2}{h_1(n) + h_2(n)} \tag{4.14}$$

The chi-square significance test rejects or accepts the null hypothesis H_0 if:

$$similarity_{texture} = \begin{cases} accept\ H_0 & if P(\chi^2|v) \leq a \\ reject\ H_0 & otherwise \end{cases}$$

where $P(\chi^2|v)$ is the χ^2 probability function, v refers degree of freedom ($number\ of\ bins - 1$) and a refers a predetermined significance level, which is empirically set to 0.05 in our case. An accepted H_0 means that both *segments* have similar texture properties.

4.4.3 Motion similarity

There are two ways of measuring the similarity between motion parameters of two *segments* [124]. The first way is to compare the motion parameters in parametric space. Motion parameters are said to be same if the distances between the same parameters are small (merging predicate). However, this approach is very sensitive to both the accuracy of the motion parameters and the merging predicate. The second way is to compare two sets of motion parameters in their residual distribution forms. The residual distribution form for a set of motion parameters $\{A, u\}$ can be represented using an error histogram generated by measuring the distance between the *image mask* of the *segment* at time t and the warped image mask of the *segment* created by applying $\{A, u\}$ to the *image mask* of the *segment* at time $t - 1$.

To measure the motion similarity between two *segments* p and q a null hypothesis, H_0,

that p moves in the same ways as q is tested. Motion of both the *segments* is said to be same if H_0 is accepted. In our motion similarity test, we have used two residual histograms ($h1$ and $h2$). The first histogram is created by computing the Euclidean distance between the *image mask* of one *segment*, say p at time t and the warped image mask created by applying the motion of p to its *image mask* at time $t-1$. The second histogram is created by computing the Euclidean distance between the *image mask* of p at time t and the warped image mask created by applying the motion of the other *segment* q to the *image mask* of p at time $t-1$. Now similar to the texture similarity measure (Section 4.4.2), the chi-square significance test is used to reject or accept the null hypothesis H_0:

$$similarity_{motion} = \begin{cases} accept\ H_0 & if P(\chi^2|v) \leq a \\ reject\ H_0 & else \end{cases}$$

where $P(\chi^2|v)$ is the χ^2 probability function, v is the degree of freedom ($number\ of\ bins - 1$) and a is a predetermined significance level, which is empirically set to 0.05 in our case. An accepted H_0 means that the motion parameters of both the *segments* are the same between time $t-1$ and t.

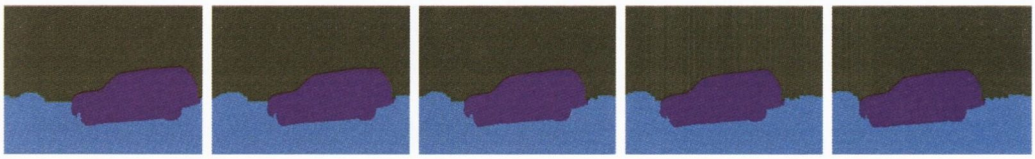
4.5 Results

In the *car1* video sequence shown in figure 4.2, a (left to right) moving car is captured by a (right to left) moving camera. Our label propagation method has divided each frame into three *segments*. It can be seen in the figure 4.3a that both the colour and the texture properties of several pixels near the boundary of the car and the road are similar. Our label propagation method, developed in chapter 3, only uses the colour and texture properties of the pixels for assigning labels. The colour and texture of the front part of the car is more similar to the neighbouring part of the road than the neighbouring parts of the car. Therefore, the label of the *segment* representing the road is assigned to a few pixels belonging to the front part of the car (Figure 4.1c). Similarly, the label of the *segment* representing the car is assigned to a few pixels belonging to the road and the background (Figure 4.1d).

Figure 4.2 shows the label propagation results of methods presented in this chapter. It is observed from the figure that accuracy and temporal consistency of results has significantly improved from the results shown in figure 4.-1. Our algorithms developed in the chapter

are able to assign correct labels to pixels belonging to the car and the road, even though the colour and the texture properties of several pixels near their common boundaries are same. Our algorithm learns the shape and the motion cues of the *segments* using the previous iteration of the label propagation. These cues helped our algorithm to archive improved results.

(a) Frames 1 – 5



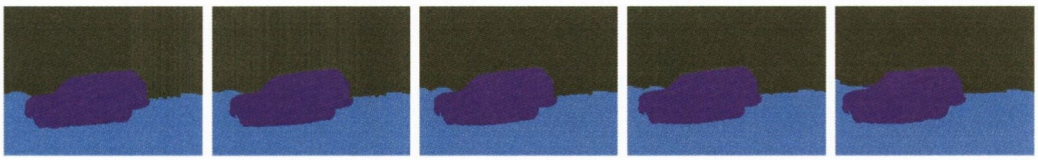
(b) Label propagation results using our method proposed in the current chapter.

(c) *Image masks* of the *segment* representing the car in segmentation results of the first frame.

(d) *Image masks* of the *segment* representing the road in segmentation results of the first frame.

(e) *Image masks* of the *segment* representing the trees in segmentation results of the first frame.

(f) Frames 6 – 10



(g) Label propagation results using our method proposed in the current chapter.

(h) *Image masks* of the *segment* representing the car in segmentation results of the first frame.

(i) *Image masks* of the *segment* representing the road in segmentation results of the first frame.

(j) *Image masks* of the *segment* representing the trees in segmentation results of the first frame.

Figure 4.2: Improved label propagation for ten frames of the *car1* video sequence of Berkeley motion segmentation dataset [107]. It can be seen that label misclassification errors present in the label propagation results in figure 4.-1 are rectified.

4.6 Conclusion

This chapter describes our contribution for improving the label propagation results for videos consisting of the rigid objects. Our algorithm learned cues, i.e. motion and *shape masks*, of the *segments* using the previous label propagation results. A label mask supporting multiple labels for each pixel was generated for the current frame. Using the energy minimisation, final labelling (single label per pixel) was achieved. The results show that the developed algorithm was able to produce accurate and temporally consistent segmentation results for the video sequences even though the colour and texture cues of the pixels belonging to the neighbouring *segments* were similar. Our algorithm was also able to assign the same label to spatially separated *segments* if their colour, texture and motion were similar.

In a video sequence, shape of *segments* can change due to relative motion between the *segments* and the camera. Learning the reasoning behind changes in shape of the *segments* can help in understanding the scene structure. In the next chapter, we present our algorithms

to learn changes in the *shape masks* of the *segments* in terms of occlusion and deocclusion.

Chapter 5

Learning the physical explanations behind changes and occlusion boundaries

Objects present in a video sequence may exhibit relative motion. Such relative motion between objects can cause occlusion or deocclusion of other objects resulting in changes in the perceived shape of these objects for an observer. Occlusion is an event where, in the view of an observer, objects disappear (completely or partially) behind other objects. Deocclusion is an event where new parts of the objects start appearing from behind the other objects. Occlusion and deocclusion change the visible shape of objects in the view of an observer. Figure 5.1 demonstrates occlusion, deocclusion and the consequent change in the shape of two *segments* in the *flower and garden* video sequence. In addition to occlusion and deocclusion other physical phenomenons such as magnification change and appearance of new objects can also change the visible shape of the objects.

In this chapter, we describe our model for categorising changes in the shape of *segments* only as occlusion or deocclusion. We refer to it as *learning the physical explanation*. Along with this, we have developed a novel algorithm for detecting occlusion boundaries present in any frame. Occlusion boundaries are boundaries of occluding *segments*. Both physical explanations and occlusion boundaries can be useful in understanding a scene structure. We present our algorithm for learning physical explanations in section 5.1 and another algorithm for detecting occlusion boundaries in section 5.2.

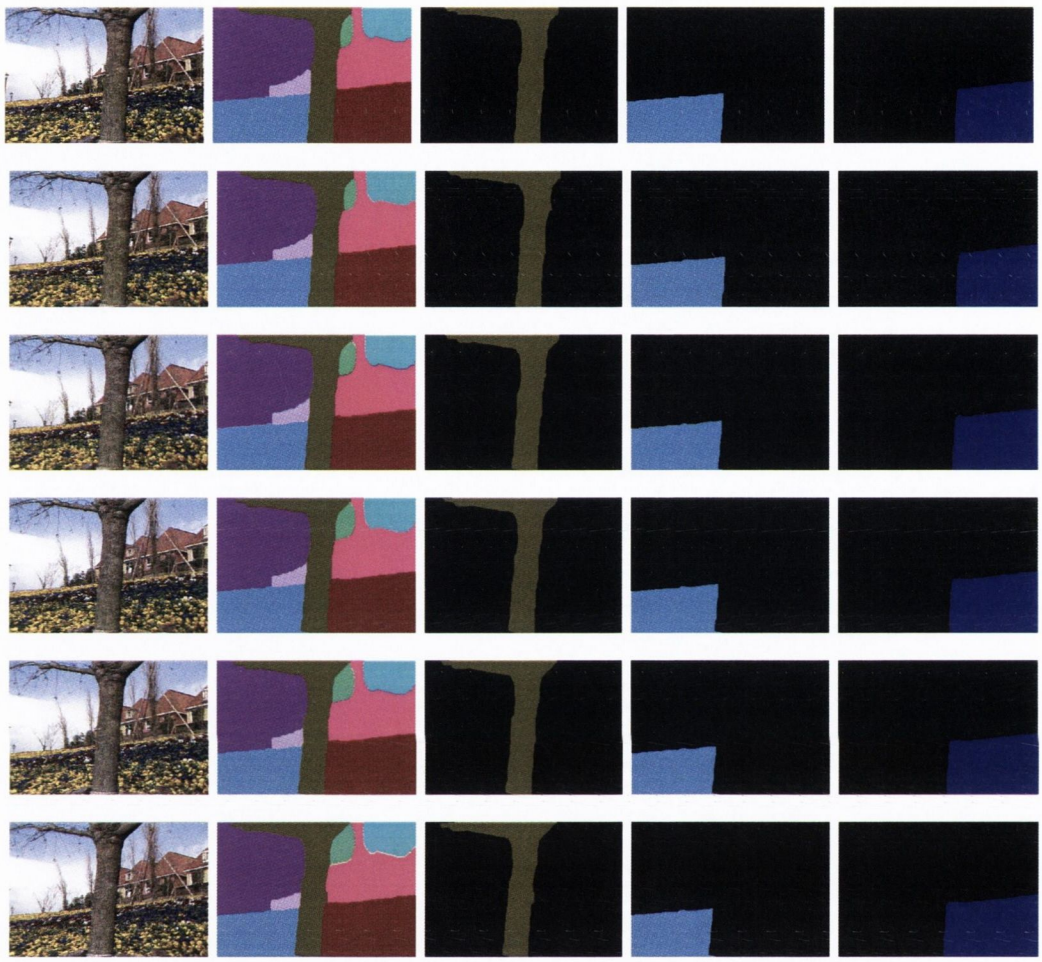


Figure 5.1: Occlusion and deocclusion in the *flower and garden* sequence. In this video sequence, the *segment* representing the tree is moving in the foreground from right to left, thus hiding parts of background objects in the direction (left) of its motion (occlusion) and revealing new parts of background objects on the right side (deocclusion). The first column shows frames from the video sequence and the second column shows our segmentation results. Our segmentation has divided the scene into eight *segments*. Columns three, four and five are showing the *label masks* of three *segments* which physically represent the tree, the flower bed to the left hand side of tree and the flower bed to the right hand side of the tree. As the *segment* representing the tree is moving from right to left, the size of the *segment* (flower bed of left side) in the fourth column is decreasing because the tree is hiding parts of this segment (occlusion). The size of the *segment* in the fifth column is growing because new parts of this segment are emerging both from right and left directions due the motion of *segment* and due the deocclusion by the *segment* representing tree.

5.1 Learning physical explanations

Any video sequence can have several moving and non-moving *segments*. These moving *segments* can occlude/deocclude other *segments* causing changes in their shape in the view plane of an observer. When a *segment* is not occluded or deoccluded by other *segments*, changes in its shape can only be explained by its own motion. In case of a occluding or deoccluding *segment*, changes in its shape may be explained by its own motion and the motion of other *segments*. We have developed a novel algorithm 7 to model changes in the shape of *segments* as occlusion or deocclusion.

Our approach for learning physical explanation and detecting occlusion boundaries (section 5.2) is similar to the layered models [125, 126, 127, 128]. Layer models interpret video sequences as layers of foreground and background *segments*. Top layers contain foreground *segments* and lower layers accommodate background *segments*. In order to divide the video sequence in layers, these models learn *shape mask* and depth ordering of each *segments* using the initial segmentation results. Final segmentation result of each frame is created by placing the *shape* masks of the *segments* on top of each other in order of their depths. Shape masks of the top layers may occlude partially or completely *shape masks* of the *segments* present in the bottom layers. Comparing the visible parts of *shape masks* of any *segment* present at two different times can be used for determining occluded and deoccluded parts of the *segment*. Moreover, occlusion boundaries can also be marked where the *shape masks* of *segments* belonging to the different layers meet.

Our algorithm uses the current *shape mask* and also current and previous *label masks* of the *segment* to explain any changes. The *shape mask*, S_t^l of any *segment* l present in the current frame I_t contains shape information of l from the first frame until the current frame. The *label mask*, M_t^l of *segment* l at time t just only contains the shape information of l present at time t. Changes in the shape of the *segment* at time t will be measured with respect to the shape of the *segment* present in a *selected* previous frame. We can call this selected frame *reference* frame I_{ref}. A *reference* frame could be any frame from the first frame of the video sequence to time $t-1$.

Algorithm 7 Learning physical explanations behind change

1: select the reference frame I_{ref}

2: **for** each segment l of the current frame at time t **do**

3: get *shape mask* S_t^l of l. S_t^l is built up from shape information of l present in all frames from time t= 1 till current time

4: get *label masks* of l from the reference frame M_{ref}^l and from the current frame M_t^l.The label mask of a segment only has the segment's shape information present in a particular frame

5: determine affine motion model parameters $(A_{ref,t}^l, u_{ref,t}^l)$ for *segment* l between reference and current frames (section 4.1).

6: create warped label mask, $M_{ref,t}^l$ by applying motion to M_{ref}^l

$$M_{\{ref,t\}}^l = A_{\{ref,t\}}^l * M_{ref}^l + u_{\{ref,t\}}^l \tag{5.1}$$

7: mark positions in S_t^l as:

$$S_t^l(x) = \begin{cases} occluded & if\ S_t^l(x) = 1\ and\ M_t^l(x) = 0 \\ deoccluded & elseif\ S_t^l(x) = 1\ and\ M_{\{ref,t\}}^l(x) = 0 \end{cases} \tag{5.2}$$

8: **end for**

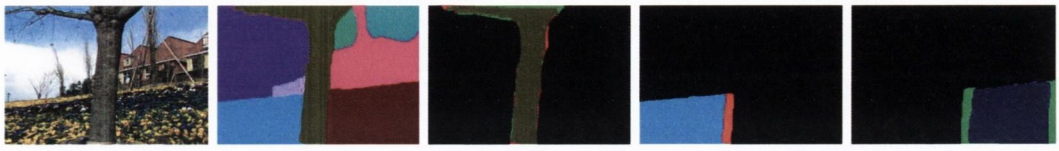
Comparison of the *shape mask* of l, S_t^l with the *label mask* from the *reference frame* M_{ref}^l can identify parts that are new (deocclusion) and comparison with the *label mask* from the current frame M_t^l can identify the parts of S_t^l that are not present in M_t^l (occlusion). *Segments* can change their positions due to relative motion between *segments* and the camera. Therefore, to compare S_t^l with M_{ref}^l the position of both masks must be aligned. Hence, instead of M_{ref}^l a warped mask $M_{\{ref,t\}}^l$ is used. This warped mask is created using the parameters $(A_{\{ref,t\}}^l, u_{\{ref,t\}}^l)$ of the affine motion model of the segment between frames I_{ref} and I_t.

Any part of *shape mask* S_t^l which is not present in the warped label mask $M_{\{ref,t\}}^l$ represents a part of the *segment* l revealed since the *reference frame*. Our algorithm marks these parts as deocclusion in S_t^l. Any part of *shape mask* S_t^l which is not present in the *label mask* M_t^l represents a part of the *segment* l hidden in the current frame.

Figure 5.2 shows the *learned physical explanations* for three *segments* of the *flower and*

garden video sequence.

(a) Frame 7



(b) Frame 11

Figure 5.2: Learning physical explanations. Row 1 shows the 7^{th} frame of the *flower and garden* sequence, segmentation result and the *shape masks* of its three (out of eight) *segments*. Row 2 shows the $\mathbf{11}^{th}$ frame of the video sequence, segmentation result and the *shape masks* of the same segments with the learned physical explanations. Our algorithm explains changes in the shape of *segments* of the $\mathbf{11}^{th}$ frame with respect to the shape of *segments* present in the 7^{th} frame. In the learned shape masks, red colour indicates occluded parts and green colour indicates deoccluded parts.

5.2 Occlusion boundaries

Occlusion occurs when, in the view of any observer, a *segment* (foreground) passes in front of another *segment* (background). Due to this foreground *segment*, background *segments* are partially or completely obscured. Boundaries present between a pair of foreground and background *segments* are called occlusion boundaries.

Information about occlusion boundaries can provide useful information about scene structure and thus can function as a strong cue for temporally consistent video segmentation. Occlusion boundary detection methods (for example [2]) mostly rely only on motion information. However, for more accurate results occlusion boundary detection methods should also use other cues such as shape information. We have developed a novel method which uses motion, shape, and edge distance information of neighbouring segments to detect occlusion boundaries in video frames. Our three stage algorithm is:

Algorithm 8 Occlusion boundary

1: Occlusion of any *segment* means that some parts of that *segment* are behind other *segments* in the current frame at time t and thus labels of the foreground *segments* are assigned to the pixels which would otherwise belong to the occluded parts. The *shape mask* of any segment represents segment's shape information accumulated from the first frame until the current frame and the *label mask* has the segment's shape information present only in the current frame. If some parts of a segment l get occluded in the current frame, these occluded pixels will not be present in its *label mask* M_t^l. These pixels will be present in the *label masks* of the foreground segments. However, occluded pixels will still be present in the *shape mask* S_t^l of the segment. Overlap between S_t^l and the *label masks* of the other *segments* present at time t can be used to distinguish between the foreground and the background *segments*. Boundaries between l and foreground *segments* are occlusion boundaries and should be associated with the foreground *segments* (Section 5.2.1).

2: At time t any deoccluded parts of segment l will only be present in its *label mask* M_t^l. Therefore, no overlap will be detected between S_t^l and the *label masks* of the other segments present at time t. In this case, the first step of our algorithm won't be able to detect any occlusion boundaries. Deoccluded parts added to S_t^l at time t represents parts which were occluded by other *segments* in previous frames. Overlap between warped shape mask $S_{t,ref}^l$ with the *label masks* of the *segments* from previous frames is used to detect occlusion boundaries in the case of deocclusion (Section 5.2.2).

3: Due to the noise present in the motion parameters our algorithm sometimes determine incorrect overlap scores and therefore, can identify false occlusion boundaries. In the last step, our algorithm uses the temporal consistency metric to identify and to remove any temporally inconsistent occlusion boundaries (Section 5.2.3).

5.2.1 Detecting occlusion boundaries using segmentation result of the current frame

In section 5.1, we presented a method to learning the reasons behind a change in the shape of segment and categorising such changes as occlusion or deocclusion. Although changes in *label mask* and *shape mask* for a given *segment* allows us to determine occlusion and deocclusion, it not directly provide information about which neighbouring *segment* has occluded and de-

occluded the current *segment*. Without knowledge of *segments* that are in the foreground it is not possible to detect occlusion boundaries. In this section, we have developed an algorithm to learn foreground *segments* for the occluded parts present in each *shape mask*. Information about foreground *segments* is further used to determine the occlusion boundaries present in the video sequence.

Algorithm 9 Overlap with *segments* of current frame

1: **for** each segment l of current frame I_t **do**
2: get shape mask, S_t^l of segment l and remove points close to its boundary using erosion. Shape mask S_t^l is built up from segment's shape information present in all frames from time t= 1 till current time
3: **for** each segment i of current frame except l **do**
4: get label mask of segment i, M_t^i. Label mask of segment i has segment's shape information present at time t
5: **if** (number of overlapping pixels in S_t^l, after removing points closer to the boundary and M_t^i are greater than a fixed percentage of the area of the smaller of *segments*) **then**
6: there must be an occlusion boundary present between *segments* l and i; and it should belongs to segment i
7: **end if**
8: **end for**
9: **end for**

In the current frame, our algorithm 9 checks for the overlap between the *shape mask* of each segment and the *label masks* of the other *segments* present in the current frame to determine the foreground segment. If the *label mask* of any segment i overlaps with the *shape mask* of segment l, it means that segment i is occluding segment l. Therefore, the boundary between l and i is an occlusion boundary and it belongs to i. The pixels marked as occluded in the shape masks of *segments* indicate that another segment is occupying these pixel positions in the current frame. Overlap (highlighted in velvet) among the *shape masks*, and *label masks* of the remaining *segments* in the current frame is used to identify foreground *segments* (Figures 5.3(e, g and i)). The pixels marked as deocclusion (highlighted in green colour) in the *shape mask* of segment indicate that other *segments* were occupying these pixel positions in the previous frames. At time t, deoccluded parts of any segment will only be present in its own *label mask*. Therefore, no overlap (Figures 5.3(k,m,o and q)) and thus no occlusion boundary (Figures 5.3(l, n, p and r)) can be detected between its *shape mask* and the *label masks* of the remaining *segments*. Figure 5.3(b) shows detected occlusion boundaries for the given frame. However, boundaries belonging to the deoccluded parts are not marked as occlusion boundaries. We have developed an extension for our algorithm to detect occlusion boundaries in case of deocclusion (Section 5.2.2).

We know that (Section 4.3) *shape masks* are created using frame to frame motion information of *segments*. Motion parameter estimation is not always noise free. Error present in the motion parameters of *segments* can lead to inaccurate *shape masks*, which can further lead to false overlaps and thus false occlusion boundaries. In our algorithm, we use two measures to deal with inaccuracies in *shape masks*. Firstly, for each shape mask we use erosion to remove points close to its boundary. Secondly, an occlusion boundary will only be present if the overlapping area between the *shape mask* (after removing points close to its boundary) and the *label mask* of another segment is more than a minimum area. Value of this minimum area depends on properties such as area and relative motion of the *segments*.

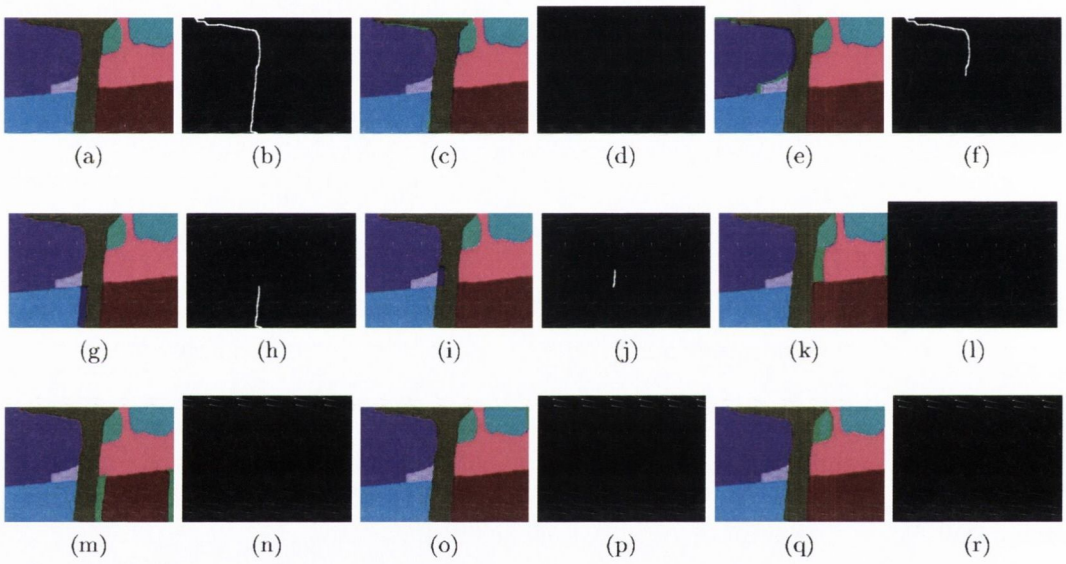


Figure 5.3: Finding occlusion boundary using label mask and shape masks at time t: (a) segmentation result of frame at time t. Frame is divided in eight *segments*; (b) occlusion boundary result of algorithm 9; (c, e, g, i, k, m, o and q) overlap between the *shape mask* of each segment with the *label masks* of remaining segments. Overlapping area is highlighted with velvet colour; (d, f, h, j, l, n, p and r) detected occlusion boundaries in each case.

In a video sequence, if the relative motion between two neighbouring *segments* is small then the overlapping area between the *shape mask* of one *segment* and the *label mask* of other *segment* will be very small. Our algorithm treats small overlapping areas as noise and hence, the boundary between overlapping *segments* will not be marked as an occlusion boundary in the current frame. The *shape masks* accumulate shape information of the *segments* present in all frames from the first frame till the current frame. Therefore, the size of the overlap between the *shape mask* of the background *segment* and the *label mask* of the foreground *segment* will increase over time and eventually our algorithm will detect an occlusion boundary between

both segments.

Algorithm 10 Overlap with *segments* of the *reference* frame

1: **for** each *segment* l of current frame I_t **do**

2: get shape mask, S_t^l of l. S_t^l is built up from segment's shape information present in all frames from time t= 1 till current time

3: **if** (l is present in frame I_{ref}) **then**

4: determine affine motion model parameters $(A_{t,ref}^l, u_{t,ref}^l)$ from frame I_t to I_{ref} for l (Section 4.1)

5: create warped shape mask, $S_{t,ref}^l$ by applying motion to S_t^i

$$S_{\{t,ref\}}^l = A_{\{t,ref\}}^l * S_t^i + u_{\{t,ref\}}^l \tag{5.3}$$

6: In $S_{t,ref}^l$, remove points close to its boundary using erosion

7: **for** each *segment* i of frame I_{ref} except l **do**

8: get *label mask* of *segment* i, M_{ref}^i. The *label mask* of any *segment* has shape information of *segment* present at any particular time

9: **if** (number of overlapping pixels in the deoccluded parts of $S_{\{t,ref\}}^l$, after removing points close to the boundary and M_{ref}^i are greater than a fixed percentage of the area of the smaller of *segments*) **then**

10: for each frame between the *reference* and the current frame, if *segments* l and i are neighbours there must be an occlusion boundary present between them; and it should belongs to segment i

11: **end if**

12: **end for**

13: **end if**

14: **end for**

5.2.2 Detecting occlusion boundaries using segmentation results of the previous frame

In the previous section, we have shown that the identity of the foreground *segment* can not be determined for deoccluded parts using the *shape masks* and the *label masks* of *segments* present at time t. Algorithm 9, does not mark boundaries between background *segments*

with deoccluded parts and foreground *segments* as occlusion boundaries. However, these boundaries are also occlusion boundaries. In this section, we present a novel algorithm to detect occlusion boundaries in the case of deocclusion. Our algorithm 10, has been developed on the fact that deoccluded parts present in the *shape mask* of any *segment* at time t were hiding behind foreground *segments* in previous frames. Consecutively, overlap between the deoccluded parts present in the *shape mask* of any segment at time t with the *label masks* of *segments* from previous frames can be used to determine these occlusion boundaries.

Both background and foreground *segments* can change their positions due to motion. Therefore, the position of both masks, the *shape mask* of the foreground *segment* and the *label mask* of the background *segment* must be aligned before comparison. Hence, instead of the *shape mask* a warped shape mask is used. This warped shape mask is created using the fitted parameters of the affine motion model of the foreground *segment* between the current frame and a previous frame. An occlusion boundary is considered to be present when the number of overlapping pixels between deoccluded parts of the warped shape mask (after removing points close to the boundary) and the *label mask* of any *segment* of the previous frame is more than a minimum area. Value of this minimum area depends on properties such as area and relative motion of the *segments*.

In a video sequence, if the relative motion between two neighbouring *segments* is small then the overlapping area between the deoccluded parts of the warped shape mask of one *segment* and the *label mask* of another *segment* will be very small. Our algorithm treats small overlapping area as noise and hence, the boundary between overlapping *segments* will not be marked as an occlusion boundary. In order to deal with this issue, our algorithm compares the *shape masks* of *segments* at time t with the *label masks* of remaining *segments* of a selected previous frame. We call this selected frame the *reference frame* I_{ref}. The *reference* frame could be any frame from the first frame of the video sequence to time $t-1$. Selection of the *reference frame* I_{ref} depends on the relative motion between the *segments*. Figure 5.4 shows our results by using frame present at time $t-2$ as the *reference frame*. Once an occlusion boundary is detected using the *shape mask* of *segments* at time t and the *label masks* of *segments* of the *reference frame*, our algorithm updates this information in all frames from the *reference* frame till the current frame.

When the overlapping area between the deoccluded parts of the warped shape mask of one *segment* and the *label mask* of another *segment* of the *reference* frame is still very small.

Our algorithm will again treat it as noise and hence, boundary between overlapping *segments* will not be marked as an occlusion boundary. This can be resolved by selecting the *reference* frame from the further back in the video.

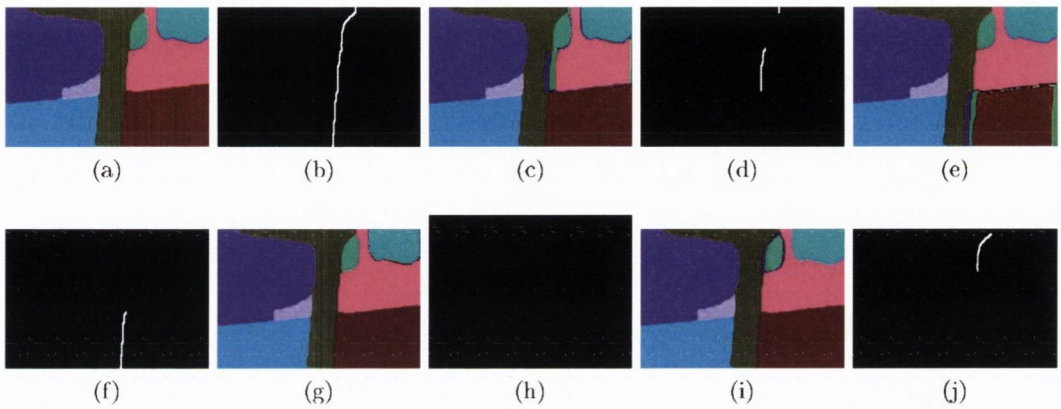


Figure 5.4: Finding occlusion boundary using shape masks present at current time t and the *label masks* of remaining *segments* of the *reference* frame: (a) segmentation result of frame at time t. Frame is divided into eight *segments*; (b) occlusion boundary result of algorithm 10; (c, e, g and i) overlap between the *shape mask* of each *segment* of time t with the *label masks* of remaining *segments* of the *reference frame*. Overlapping area is highlighted in velvet colour; (d, f, h and j) occlusion boundaries detected in each case.

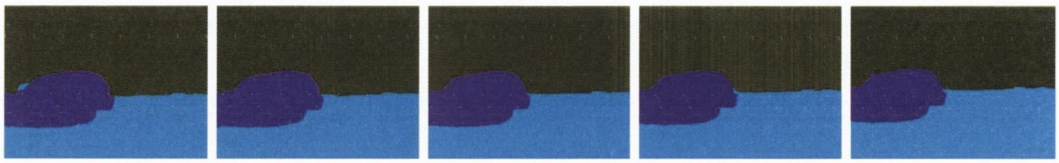
5.2.3 Updating occlusion boundaries using temporal consistency metric

In the previous sections, our algorithm detected the presence of occlusion boundaries by checking the overlaps among segments' *shape masks* and *label masks* present both in the current frame and in the previous frame.

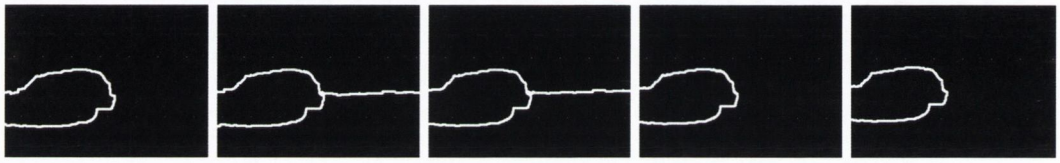
A boundary marked as an occlusion boundary should remain an occlusion boundary for at least a number of frames. In this step, we compute temporal consistency score for the occlusion boundaries of each frame using our temporal consistency metric (Section 6.3). We have used a block of five frames to compute the temporal consistency. Boundaries for which the temporal consistency score is zero (only present in 1 or 2 frames) are removed from the occlusion boundary masks of those frames. However, a false occlusion boundary cannot be identified if it is temporally consistent (present in number of frames more than the number of frames in block).

Figure 5.5 shows removal of an occlusion boundary due to a poor temporal consistency score. A boundary between the *segments* representing road and sky is determined as an occlusion boundary in two frames, **16** and **17**. This boundary is not marked as occlusion

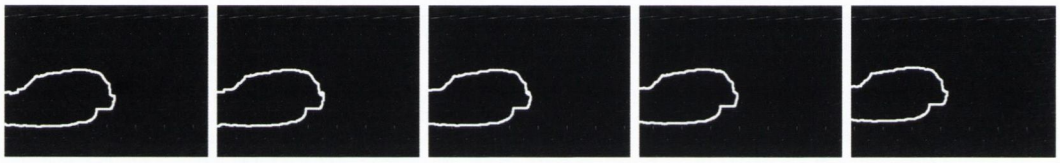
(a) Frames 15 – 17



(b) Segmentation results



(c) Occlusion boundaries



(d) Occlusion boundaries after removing temporally inconsistent occlusion boundaries.

Figure 5.5: Removing temporally inconsistent occlusion boundaries (*car1* video sequence of Berkeley Motion segmentation dataset [107]).

boundary in any other frame. Our temporal consistency metric assign zeros consistency score for all pixels belonging to this boundary. Therefore, this boundary is removed from final occlusion boundary masks.

5.3 Results

Figures 5.6 and 5.7 show segmentation results, the *learned physical explanations* and the occlusion boundaries for **10** frames from the *flower and garden* video sequence and the *car1* video sequence from Berkeley motion segmentation dataset [107].

The *flower and garden* video sequence shown in figure 5.6 is captured by a moving camera and therefore, the relative positions of the objects also changed between video frames. Our label propagation methods starts with the manual segmentation of the first processes frame where scene is divided into eight *segments* (rows **2** and **6**). Due to camera motion the *segment* representing the tree moves in the foreground from right to left, thus hiding parts of the *segments* in the direction (left) of its motion (occlusion) and revealing new parts of the

segments on the right side (deocclusion). Our algorithm 7 is able to detect these changes successfully (rows 3 and 7) for all the *segments* except one which is present at the top left position in the manual segmentation of the first frame. In each frame, occluded and deoccluded parts of the background *segments* are highlighted in red and green colours respectively. New parts appearing from right side of the frames due to the relative motion between background *segments* and camera are also marked as deocclusion. Rows 4 and 8 show detected occlusion boundaries.

Accuracy of both the learned physical explanations and the occlusion boundaries depends on the accuracy of the fitted motion parameters of *segments*. In figure 5.6, the *segment* present in the top left position in the segmentation results is under segmented. A single label is assigned to the pixels belonging to the sky and branches of the tree. For this *segment*, relative motion between tree branches and camera is the dominating motion and hence, gets selected as the motion of the *segment*. Incorrect motion information leads to a false deoccluded area (with the small segment below it), and consequently a false occlusion boundary is detected between the top left *segment* and the *segment* (highlighted in pink) below it. Moreover, for the same reason a small occluded area is detected between the *segment* at the top left and other *segment* present at the bottom left. However, the boundary between these *segments* is not marked as occlusion boundary because overlapping area is smaller than a fixed percentage (0.05%) of the area of the smaller of *segments*.

In the *car1* video sequence shown in figure 5.7, a moving (left to right) car is captured by a moving (right to left) camera. Our segmentation method has divided each frame into three *segments*. The *segment* representing the car is moving in the foreground from right to left, thus hiding parts of the *segments* in the direction (left) of its motion (occlusion) and revealing new parts of the *segments* on the right side (deocclusion). Due to rich colour and texture properties, the motion detection method (Section 4.1) is able to estimate the correct motion parameters for the *segments* representing the car and trees. Therefore, our algorithm successfully detected occluded and deoccluded parts of these segments. For the *segment* representing the road, the accuracy of the learned physical explanations suffers due to the poor motion parameter estimation (rows 3 and 7). Due to the camera motion, new parts of the road appears from the left side and our algorithm fails to recognise these parts.

Occlusion boundaries are detected only using the segmentation results of video frames. In the case of incorrect segmentation, where boundaries of any segment do not represent

real physical boundaries, our algorithm may mark false occlusion boundaries. In figure 5.7, segmentation assigns a single label to the pixels belonging to the car and the shadow of the car (rows 2 and 6). Our occlusion boundary detection algorithms detect occlusion between the *segment* representing car and shadow; and the *segment* representing road (rows 4 and 8). Therefore, the boundary present between the *segments* are marked as an occlusion boundary. In reality, the shadow part belongs to the road and hence, the marked boundary is not an occlusion boundary.

Our algorithms for learning the physical explanations and occlusion boundaries use shape and motion cues of the *segments*. These cues are learnt using the segmentation results of the video frames. Accuracy of the learned physical explanations and occlusion boundaries depends on the accuracy of shape and motion cues. Consequently, depends on the accuracy and the temporal consistency of the segmentation results. Results presented in figures 5.6 and 5.7 show that algorithm is able to learn the physical explanations and occlusion boundaries correctly if motion and shape cues are correct. Our algorithms for learning the physical explanations and occlusion boundaries do not require label propagation algorithms presented in chapters 3 and 4. Only segmentation result is required as an input. Input segmentation result can be achieved using any state of the art method, e.g, α-expansion [16] and methods proposed by Wang and Collomosse [1]. In the other words, algorithms proposed in this chapter can be applied to the segmentation results achieved from any state of the art video segmentation method.

5.4 Conclusion

This chapter describes our contribution for *learning the physical explanations* behind changes and occlusion boundaries. The developed algorithms used the *shape masks* and motion cues learned in the previous chapter to model any change in the shape of a *segment* in the current frame as occlusion or deocclusion. Any part of the *segment* in the current frame which was not present in previous frames is a new part and marked as deocclusion. Any part of the *segment* that is not present in the current frame but was present in the previous frame is marked as occlusion. Learning these physical explanation behind changes was further used by our algorithm for identifying the occlusion boundaries.

After segmenting frames of the video sequence, segmentation methods should measure

the performance of segmentation quantitatively. In next chapter, we present our developed metrics to measure the accuracy and the temporal consistency of segmentation results.

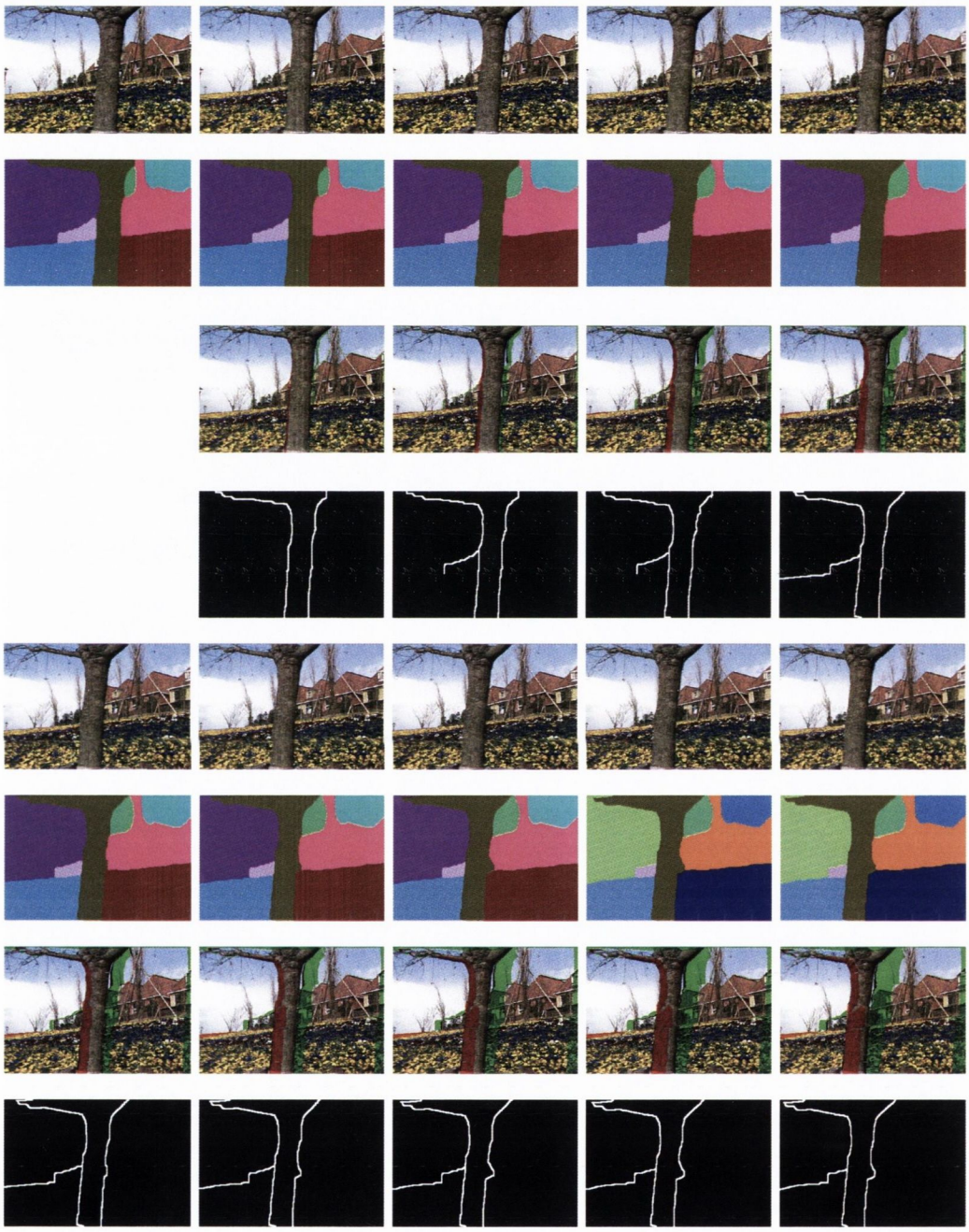


Figure 5.6: The learned physical explanations (rows 3 and 7) and occlusion boundaries (rows 4 and 8) for ten frames (rows 1 and 5) of the *flower and garden* video sequence. For each frame any change in shape of its *segments* is detected using the shape of *segments* present in the first frame. The *label masks* of frames are shown in rows 2 and 6.

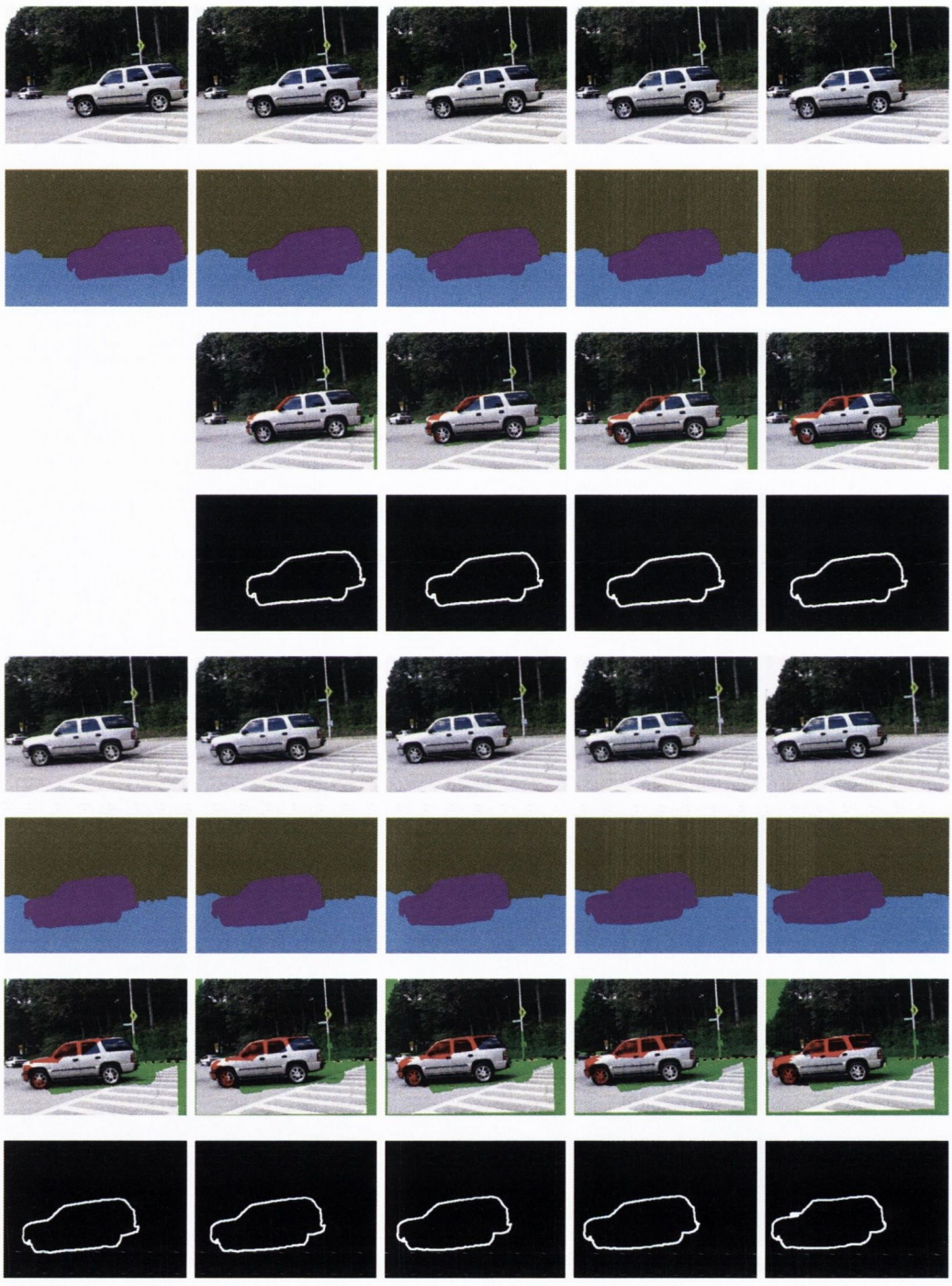


Figure 5.7: The learned physical explanations (rows 3 and 7) and occlusion boundaries (rows 4 and 8) for ten frames (rows 1 and 5) of the *car1* video sequence of Berkeley Motion segmentation dataset [107]. For each frame any change in shape of its *segments* is detected using the shape of *segments* present in the first frame. The *label masks* of frames are shown in rows 2 and 6.

Chapter 6

Evaluation metrices

We have developed three metrics: spatial consistency, size consistency and temporal consistency metrics to measure the accuracy and temporal consistency of the segmentation results. In this research, we use human annotated ground truth data to measure the accuracy of our frame/image segmentation results but do not require any ground truth to assess video temporal consistency. As we have stated earlier (Chapter 1), segmentation is an ill posed problem where segmentation of an image/frame depends on the annotator's knowledge about the scene. Human annotators use very high levels of information for segmentation. This level of segmentation is tremendously difficult to achieve for any of the segmentation algorithms, as it requires object level knowledge and understanding. Therefore, a single region (object) present in the ground truth normally gets segmented as several regions by the segmentation algorithm (over-segmentation). Figure 6.1 shows an image, its ground truth image and its segmentation result. It can be seen that the human annotator segments the hat as one region. However, the segmentation method creates two segments for the hat. This kind of segmentation is called over-segmentation. Segmentation accuracy metric should not penalise segmentation for over-segmentation. However, it must consider the numbers and sizes of the over segmented regions to measure the segmentation results accurately. We have developed two metrics spatial consistency and size consistency detailed in sections 6.1 and 6.2 for evaluation of the accuracy of frame/image segmentation.

Segmentation of an object is said to be temporally consistent if each part/pixel of the object gets the same label over the whole video sequence. Ground truth with pixel to pixel correspondence information would be required to measure the temporal consistency of the segmentation. However, creating such detailed ground truth is a very difficult and labour

(a) Original image

(b) Ground truth image from Berkeley Database[4].

(c) Segmented image using NCut [26].

Figure 6.1: Ground truth and over-segmented results, Spatial Consistency:- 0.65, Size Consistency:- 0.43

intensive task. We have developed a novel metric that can measure the temporal consistency of the segmentation results without using detailed ground truth. This is detailed in section 6.3. Our method measures the temporal consistency of segmentation results of the current frames using only the boundary pixels of past and future segmentation results. Currently, no robust approach for measuring the temporal consistency with out ground truth is present in the research literature.

6.1 Spatial consistency metric

The spatial consistency metric measures the accuracy of the segmentation against the ground truth. Ground truth segmentation is a high level (object based) segmentation and comparatively over-segmentation is expected as the result. Therefore, over-segmented regions should be clustered together before the comparison with objects in the ground truth. In the first step, our method cluster *segments* having maximum spatial overlap with the same ground truth object. Each such cluster is called an *evaluation set.* An *evaluation set*, E_i holds any *segment* which has maximum spatial overlap with the ground truth object, G_i.

In the second step, *false positives* and *false negatives* are used to compute spatial accuracy. A *false negative* refers to the case when a pixel p of G_i is not present in E_i. A *false positive* refers to the case when a pixel p of E_i is not present in G_i. Spatial consistency score for *segments* of E_i can be expressed as:

$$\textit{Spatial consistency score of } E_i = 1 - \left(w \times \frac{\text{total number of } \textit{false negatives}}{\text{total number of pixels in } G_i} + (1 - w) \times \frac{\text{total number of } \textit{false positives}}{\text{total number of pixels in } E_i} \right) \tag{6.1}$$

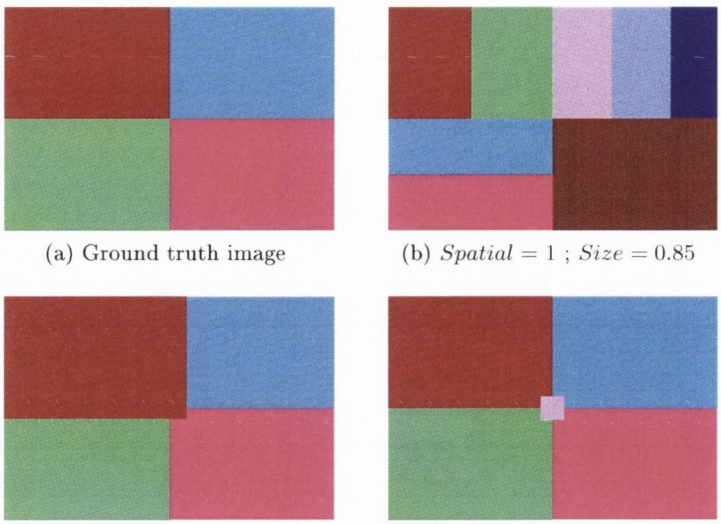
where, w is weight parameters with value always less than or equal to one ($w \leq 1$). Finally spatial consistency score for a frame can be given as weighted sum of spatial consistency scores of *evaluation sets*:

$$\textit{Spatial consistency score of a frame} = \frac{\sum_{i=1}^{n} \text{total number of pixels in } E_i \times \text{Spatial consistency score of } E_i}{\text{total number of pixels in frame}} \tag{6.2}$$

where, n is total number of *evaluation sets* present in the frame. The ratio of the number of pixels present in an *evaluation set* over total number of pixels allows bigger *evaluation sets* to contribute more to the spatial consistency score of the frame.

6.2 Size consistency metric

The spatial consistency metric depends only on the number of false positives and false negatives values. These false positive and false negative values are computed after clustering the *segments*. The spatial consistency metric does not consider total number and individual area of the *segments* of E_i. In a specific case, lets assume a segmentation algorithm divides an image by assigning a unique label to each pixel. As a result, each *segment* is of size 1 pixel. Our spatial consistency metric will cluster the *segments* to create evaluation sets without considering their sizes and numbers. Using equation 6.2 this segmentation will get score 1 for the spatial consistency. Therefore, the spatial consistency metric is not enough to measure the segmentation quality. Hence, we introduce another metric called *size consistency* to penalise the segmentation quality based on the number and size of *segments* present in each cluster.



(a) Ground truth image

(b) *Spatial* = 1 ; *Size* = 0.85

(c) *Spatial* = 0.9043 ; *Size* = 1

(d) *Spatial* = 0.9887 ; *Size* = 0.9841

Figure 6.2: Examples of spatial and size consistency scores using different synthetic data. Value of w is set to 0.5 in the spatial consistency metric computation.

Score of our *size consistency* metric for *segments* of E_i can be given as:

$$Size\ consistency\ score\ of\ E_i = \begin{cases} 1 & n = 1 \\ 1 - \frac{1}{n} \times \sum_{l=1}^{n} weight_l \times sizePenalty_l & otherwise \end{cases} \tag{6.3}$$

where, n is the total number of segments present in evaluation set E_i. For each *segment* l of E_i, values of *sizePenalty* and *weight* can be calculated as:

$$sizePenalty_l = \frac{\text{total number of pixels in } G_i - \text{total number of pixels in } l}{\text{total number of pixels in } G_i + \text{total number of pixels in } l} \tag{6.4}$$

and,

$$weight_l = \frac{\text{total number of pixels in } l}{\text{total number of pixels in } E_i} \tag{6.5}$$

where, G_i is object in the ground truth. Variable weight ensures that metric penalise each evaluation set E_i according to the relative size and numbers of its member *segments*. Finally, size consistency for segmentation results of a frame can be expressed as:

$$Size\ consistency\ score\ of\ a\ frame = \frac{\sum_{i=1}^{n} \text{Size consistency score of } E_i}{n} \tag{6.6}$$

where, n is total number of *evaluation sets* present in the frame.

Figure (6.2) displays the spatial and the size consistency scores for different scenarios using synthetic data.

6.3 Temporal consistency metric

As defined earlier, temporal consistency requires that pixels belonging to each object should get the same label over the entire video sequence. Any change in the labelling of pixels belonging to an object should be considered as a breakdown of the temporal consistency associated the object. Generally in a video, objects change their shapes and positions due to relative motion between the camera and the objects. As a result, shapes and positions of the respective *segments* should change in the segmentation results. We have developed a novel algorithm to measure the temporal consistency score of any *segment* by analysing how the segment is changing its shape and position over the time. If the change in shape or position of any *segment* is consistent over a period of time we infer that the segment is temporally consistent.

In our developed method, temporal consistency of *segments* of the current label mask is computed using the segmentation results of the past and the future frames. For this purpose a *block* is constructed using equal numbers of the past and the future segmentation results with segmentation results of the current frame at the centre. Only boundary pixels are used to measure the temporal consistency of *segments*. Temporal consistency score of a *segment* is the average of temporal consistency scores of its boundary pixels:

$$\begin{aligned} &\textit{temporal consistency score of segment } l \\ &= \frac{\sum_{b=1}^{m} \textit{temporal consistency score of boundary pixel } b}{m} \end{aligned} \tag{6.7}$$

where m is the total number of boundary pixels present in *segment* l.

Temporal consistency score for the boundary pixels of the current label mask is computed using the segmentation results of the past and the future frames (Section 6.3.1). A video frame is consists of *segments* so its temporal consistency score can be computed as weighted

sum of temporal consistency scores of its *segments*:

$$\begin{aligned} &\textit{temporal consistency score of a frame} \\ &= \frac{\sum_{l=1}^{L} M_l \times \textit{temporal consistency score of segment } l}{N} \end{aligned} \tag{6.8}$$

where M_l represents total number of pixels in *segment* l, L is the total number of *segments*, and N is total number of pixels present in the frame.

6.3.1 Temporal consistency score of boundary pixels

The temporal consistency score for the boundary pixels of the current label mask is computed using a *block* of n label masks from time $(t - \lfloor \frac{n}{2} \rfloor) \rightarrow (t + \lfloor \frac{n}{2} \rfloor)$ with the current label mask at the centre. A block is consists of odd number of frames therefore the floor function ($\lfloor\ \rfloor$) is used to get the number of frames present in the forward and the backward directions. A boundary pixel is said to be temporally consistent if and only if labels of all n corresponding pixels are same. Otherwise such pixel is said not be temporally consistent.

If we knew the ground truth with pixel to pixel correspondence, measuring the temporal consistency would be an easier task. Correspondence of pixel over the block can give information about pixel at n different time instances. However, getting such ground truth is a non trivial task. In our developed method (Algorithm 11), collection of such highly detailed ground truth is not required. We have developed a method to estimate the correspondence for the boundary pixels (Section 6.3.2). For each boundary pixel of the current label mask our method returns several possible sets of corresponding pixels from each label mask of the block. Any set of such possible corresponding pixels, one from each label mask is called a *path*. The temporal consistency score of each *path* score is computed as described in section 6.3.3. A *path* with the maximum consistency score can be considered as the *path* having the best set of corresponding pixels thus its score is used as temporal consistency score of the pixel in the block.

Algorithm 11 Temporal consistency score of boundary pixels

1: **for** each boundary pixel b of *segment* l of the central frame **do**

2: create paths from possible corresponding pixels present in the frames of the block (algorithm 12)

3: **if** path exists **then**

4: for each path compute the temporal consistency score (algorithm 13) and update temporal consistency scores for all the boundary pixels present in the path if boundary pixels are not *marked* (line 12) and the new score is greater than the existing scores

5: **else**

6: set temporal consistency score of b to 0, the minimum possible consistency score, if no score is assigned to b.

7: **end if**

8: **end for**

9: **for** each *segment* l of the current frame, which is not present in the previous frames (new *segment*) or in the next frames (*segment* is disappearing in the next frames) of the block **do**.

10: update score of each boundary pixel:

$$= \begin{cases} 1 & if\ width_l \leq 2 * d_{max} + 1\ or\ height_l \leq 2 * d_{max} + 1 \\ zero & otherwise \end{cases}$$

where d_{max} is the spatio temporal displacement constraint and, $width_l$ and $height_l$ are the width and the height of the smallest bounding box holding the *segment* l.

11: **if** score of boundary pixels is not updated in the previous step **then**

12: *mark* all boundary pixels belonging to l to avoid any change in their scores in future

13: **end if**

14: **end for**

A *path* with higher temporal consistency score can set scores of corresponding boundary pixels in both the past and the future segmentation masks. This property is very useful in case of dealing with occlusion and deocclusion because in both cases the occluding or the deoccluding boundary pixels have consistent motions in at least one direction. In our algorithm, the temporal consistency score of the boundary pixels of the central frame of the

current block will keep changing until the current central frame becomes the first frame of a future block.

For a boundary pixel of the central frame our algorithm creates no *path* if there are no corresponding pixels present in the previous or in the next frames of the block. In such cases, the minimum temporal consistency score of zero will be assigned to the boundary pixel (line 6). In a video sequence, new objects can appear over the period of the time. For boundary pixels of the *segments* representing these objects our algorithm will not create any *path* because of missing correspondence in the backward direction. Consequently, the minimum score will be assigned to these boundary pixels as well.

To allow appearance of new object a post processing step is added in the algorithm (lines 9 -14). In the first step, our algorithm detects the *segments* for which no correspondence is present in either of the directions (line 9). These *segments* can belong to new objects (no corresponding pixel present in the backward direction) or disappeared objects (no corresponding pixel present in the forward direction). Our algorithm treats appearance and disappearance of any *segment* temporally consistent if size of the *segment* in the central frame of the block is within the allowed limit. Using spatial temporal displacement constraint d_{max} we know that the distance by which the *segment* can move between frames is limited. Therefore, width or height of the new *segment* or the disappeared *segment* should not be more than $2 * d_{max} + 1$. In the next step, our algorithm updates the score of boundary pixels of detected *segments*, if their heights or widths are less than $2 * d_{max} + 1$. Otherwise, the *segment* is treated as temporally inconsistent and score of its boundary pixels will not be updated in future.

6.3.2 Creation of *paths*

A *path* includes one possible corresponding pixel from each label mask of the block. As a result, length of a *path* (number of pixels in the path) should be equal to the number of masks in the block.

Using spatio-temporal displacement constraint (Section 3.3) we already know that any *segment* present in the current frame can move by a maximum d_{max} pixels in the next frame. The same constraint applies for the boundary pixels as well. Therefore, boundary pixel p of segment l present in (x, y) position in the current frame must be present within the d_{max} radial distance from (x, y) position in both the previous and next frames. In next frames, the same pixel must be present within $2 * d_{max}$ distance in any direction (Figure 6.3).

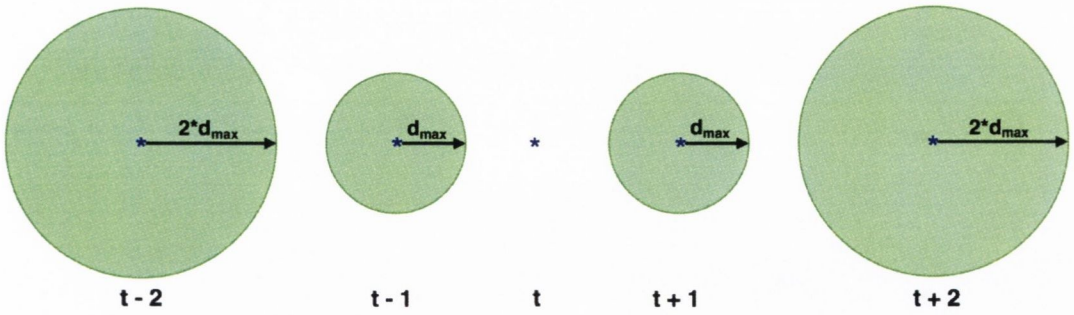


Figure 6.3: Spatio-temporal neighbourhoods of pixel ($\star$) in backward and forward directions at times $t-2$, $t-1$, $t+1$ and $t+2$. Spatio-temporal neighbourhood is highlighted in green and arrows show their lengths.

In our method, knowledge of accurate pixel correspondence is not required. Several *paths* are created for the current boundary pixels taking pixels present in their spatio temporal neighbourhoods one by one (Algorithm 12). The number of paths depends on the number of frames in the block and the number of boundary pixels present in spatio temporal neighbourhoods. For a block of n frames, if M is the maximum number of boundary pixels that may be present in the spatio temporal neighbourhood of a boundary pixel b in the next frame, M^{n-1} is the maximum number of paths that can be created for b.

Figure 6.4 shows a few paths created for a boundary pixel b, shown as $\star$ in the figure, in the forward direction. Lets assume that M is the number of boundary pixels present in spatio temporal neighbourhood of b at time $t+1$. A maximum of M paths may pass through b at time t in the forward direction. For each boundary pixel b_{t+1} present in the spatio temporal neighbourhood of b at time $t+1$, there are M boundary pixels present in its spatio temporal neighbourhood at time $t+2$. Again, M is the maximum number of paths that may pass through b_{t+1} at time $t+1$ in the forward direction. For each of the M paths, exist between time t and $t+1$, maximum M paths may exist between time $t+1$ and $t+2$, consequently maximum $M \times M$ paths may exist between time t and $t+2$. In the generalised form, there may be maximum $M^{\lfloor \frac{n}{2} \rfloor}$ paths exist in forward direction for a block of frames of length n. Similarly, there may be maximum $M^{\lfloor \frac{n}{2} \rfloor}$ paths exist in the backward direction. For each path of the backward direction there may be maximum $M^{\lfloor \frac{n}{2} \rfloor}$ paths exist in the forward direction. Therefore, $M^{n-1} = (M^{\lfloor \frac{n}{2} \rfloor} \times M^{\lfloor \frac{n}{2} \rfloor})$ may the maximum number of paths exist for a boundary pixel of the central frame in the block of n frames.

Algorithm 12 Create paths

1: for each boundary pixel p of segment l present in the central frame I_t, path is a set of n corresponding boundary pixels, one from each of the frames present in the block:

2: $path = \{$ *boundary pixel* p_i *of segment* l *in frame* I_i *:* p_i *is present within spatio temporal displacement distance from* p*'s position in the central frame,* $i = t - \frac{n}{2} \ldots t - \frac{n}{2}\}$

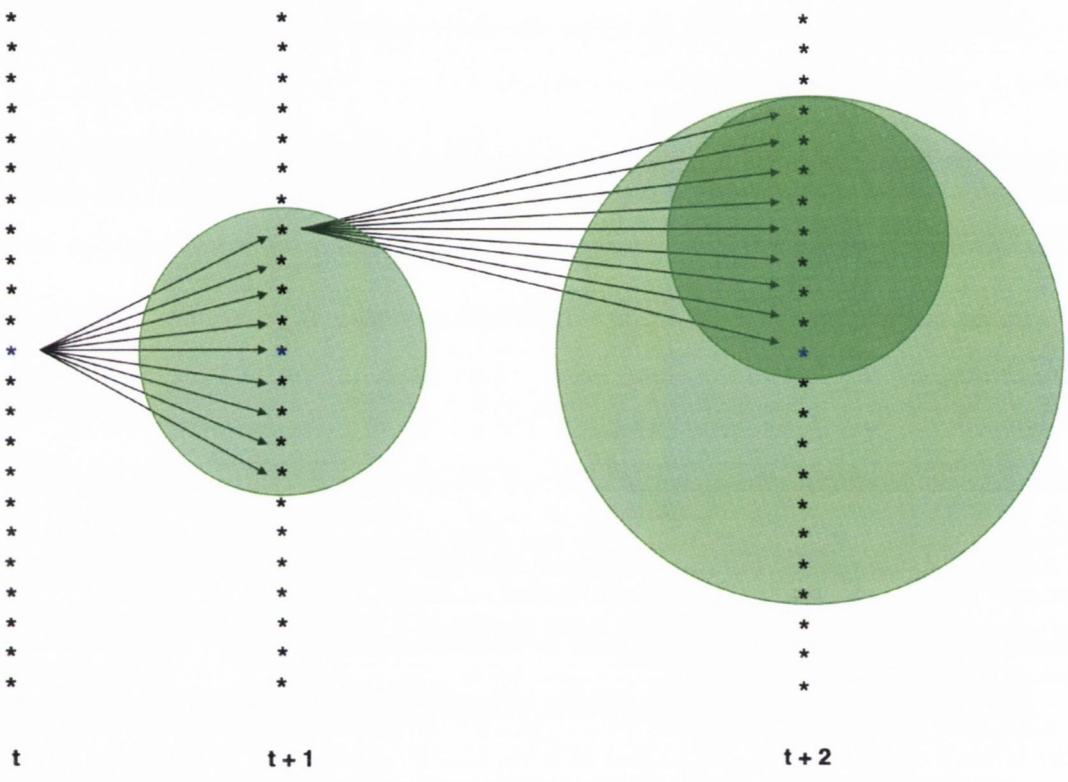


Figure 6.4: *Paths* for pixel ($\star$) in the forward direction at times $t+1$ and $t+2$. Spatio temporal neighbourhood is highlighted in green and arrows show corresponding pixels in spatial temporal neighbourhoods.

6.3.3 Temporal consistency score of a *path*

In the previous section, we have created several *paths* for each boundary pixel of the current frame. A *path* holds the boundary pixel's positions in the previous and the future frames. In this section, we have developed a novel method to measure the temporal inconsistency score for each *path*.

Algorithm 13 Compute path score

1: fit curves separately in X and Y data points with respect to the time t.

2: compute residual at time$i = (t - \lfloor n/2 \rfloor) \rightarrow (t + \lfloor n/2 \rfloor)$ in X direction:

$$r_i^x = x_i - \hat{x}_i \tag{6.9}$$

and in Y direction

$$r_i^y = y_i - \hat{y}_i \tag{6.10}$$

where x_i and y_i are the actual position of the point at time i and $\hat{x}_i$ and $\hat{y}_i$ are the values of the curves at time i.

3: compare residuals with the residual threshold r_T:

$$r_i^x = \begin{cases} r_i^x & if\ r_i^y > r_T \\ 0 & otherwise \end{cases}$$

and,

$$r_i^y = \begin{cases} r_i^y & if\ r_i^y > r_T \\ 0 & otherwise \end{cases}$$

4: inconsistency score of path is normalised sum of the residuals at each point:

$$= \sum_{i=t-\lfloor n/2 \rfloor}^{t+\lfloor n/2 \rfloor} \frac{\sqrt{(r_i^x)^2 + (r_i^y)^2}}{d_{max}^i} \tag{6.11}$$

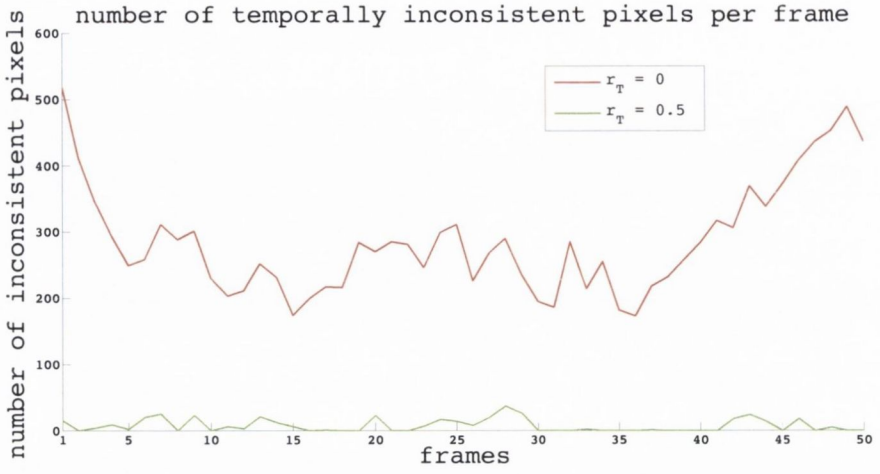
where d_{max}^i is maximum possible distance (spatio temporal displacement constraint) that any boundary pixel of the central frame is allowed to move in frame I_i.

5:

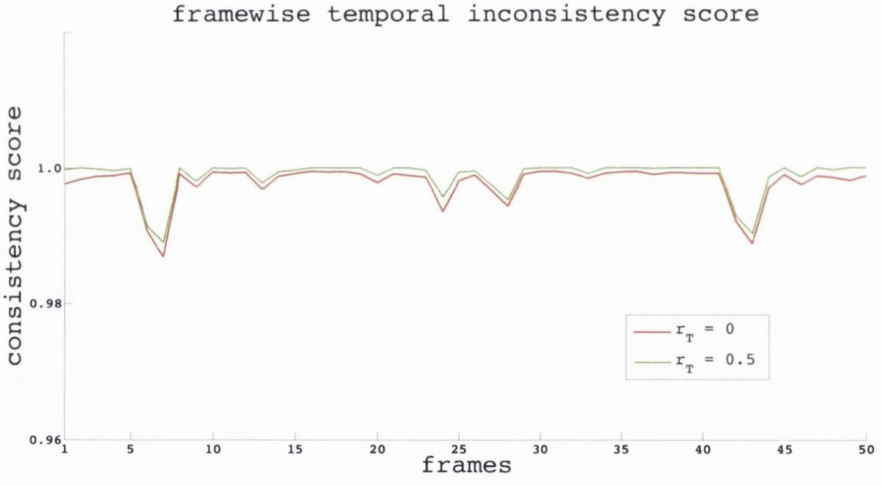
$$temporal\ consistency\ score\ of\ path = 1 - temporal\ inconsistency\ score\ of\ path \tag{6.12}$$

In a video, *segments* can change their shapes and positions either due to the relative motions between *segments* and camera or due to the segmentation errors. If the change is only due to the relative motions, trajectories of the boundary pixels should follow any standard curve, e.g., parabolic or sinusoidal. Therefore, to check temporal consistency of the *segments*

it is wise to fit the trajectories of boundary points (*path*) of the *segments* with standard curves. A smooth fit will imply temporally consistent trajectory whereas residuals can be treated as temporal inconsistencies.



(a) Number of inconsistent pixels



(b) Temporal consistency

Figure 6.5: Temporal consistency scores for 50 frames of the *boy* video sequence. For measuring the temporal inconsistency scores we have used block of five frames; quadratic curves with residual thresholds 0 and 0.5; and value of spatio-temporal displacement constraint is set to 10.

Our developed algorithm 13 fits curves to the data points present in the *path*. A *path* holds $2D$ positions (X, Y) of a boundary pixels at different times. Therefore, curve fitting has to be done in $3D(XY + time)$. For simplicity we have broken down the problem into $2D$ by checking how trajectories of the boundary pixel are changing in X and Y dimensions separately. Our algorithm fits two separate curves one each to X and Y values, respectively. The curves must

pass through the central boundary point of the path. After fitting the curves, residuals are measured at each point to measure the divergence of the point from the ideal path. Finally, temporal inconsistency score of the *path* is given by the normalised sum of squared residuals of both the curves. Spatio temporal displacement constraint is used for normalising because spatio temporal displacement constraint is the maximum possible distance any pixel can move in the next frame and therefore, value of residual cannot be more than this constraint.

In our algorithm, we have used a threshold called *residual threshold* (r_T) to neglect the segmentation error (residual) up to a fixed value. Figure 6.5 shows frame wise temporal inconsistency score and number of inconsistent pixels for our segmentation result of the *boy* video sequence. To measure the temporal inconsistency score we have used block of five frames, quadratic curves with residual thresholds (r_T) 0 and 0.5 and value of spatio temporal displacement constraint is set to 10. Number of inconsistent pixels drops significantly with higher value of the residual threshold. It is also clear from the figure that for $r_T = 0.5$ value of temporal inconsistency is close to zero for most of the frames.

6.4 Results

In this section, we evaluate the working of our spatial, size and temporal consistency metrics with the help of our segmentation results and the results obtained from the α-expansion method proposed by Boykov et al. [16]. The result demonstrates that our metrics are able to detect and penalize spatial inaccuracy and temporal inconsistency present in the video segmentation results.

To measure the spatial inaccuracies our spatial and size consistency metrics compare the segmentation results with the ground truth. Figure 6.6 shows ground truth data of five key frames of the *boy* sequence and segmentation results. Both ground truth data and video sequence are parts of video data presented by Wang et al. [1]. It can be seen that in comparison to the ground truth, several small *segments* are present in the segmentation results of the α-expansion method. Our spatial consistency metric merges the *segments* before comparing them with the ground truth. Therefore, appearance of new *segments* and region merging/splitting do not have a significant impact on the spatial consistency score. However, our size consistency metric is able to handle these cases. Table 6.1 shows the scores of our spatial and size consistency metrics. Our segmentation results are not over-segmented. As a

(a) Five key frames

(b) Ground truth

(c) Label propagation results using the α-expansion [16].

(d) Our label propagation results

Figure 6.6: Ground truth and segmentation results of α-expansion and our method for five frames of the *boy* sequence. Both α-expansion and our label propagation method start with the same manually segmented results of the first processed frame (Figure 3.11). Table 6.1 shows the spatial and the size consistency scores of segmentation results of both the methods for these frames. In comparison with the ground truth, α-expansion method has changed labels of many pixels completely. For example, the same label is assigned to most of the pixels belonging to the trees. The spatial consistency metric combines the *segments* before comparison with the ground truth. Therefore, label change of the *segments* does not change the score of spatial consistency metric significantly. Size consistency score of α-expansion method suffers due to the presence of several small *segments* in the results. Spatial consistency of our results suffer due to incorrect labelling of few pixels along the border of the *segment* representing the ground and the *segments* representing the house and the trees and the bush. Few pixels belonging to the ground *segment*, near the feet of the boy, are also incorrectly labelled as the *segment* representing the trouser and the gloves. Size consistency score for our segmentation results is 1 due to absence of over-segmentation for all five frames.

result, size consistency metric assigns the best possible score of 1 to segmentation results of all the five frames.

		frames				
		1	2	3	4	5
Spatial consistency score	α-expansion	0.8173	0.8170	0.8190	0.7732	0.7954
	our method	0.8895	0.8999	0.8995	0.8724	0.8903
Size consistency score	α-expansion	0.8064	0.7697	0.8110	0.8238	0.7853
	our method	1	1	1	1	1

Table 6.1: Spatial and size consistency scores for frames shown in figure 6.6. Value of variable w is set to 0.5 in the spatial consistency metric computation (Equation 6.1).

Our temporal consistency metric does not require ground truth for evaluating changes in the segmentation results over time. This metric should penalise segmentation results for any inconsistent change, such as sudden appearance and disappearance of *segments*. Also it should not penalise any consistent change, such as occlusion and deocclusion. Figure 6.7 shows temporal consistency scores for 50 frames of the *boy* video sequence. To measure the temporal consistency of each frame, a block of five frames is created and quadratic curves are used to find out the best corresponding pixels. The residual score of the best-fit curve (path) is used as temporal inconsistency score. The segmentation results of α-expansion method changes substantially due to sudden appearance and disappearance of *segments*. We observed that the temporal consistency metric has penalised the segmentation results of the α-expansion method more than our segmentation results.

Figure 6.8 shows segmentation results for 5 sequential frames (17 - 21) of the *boy* sequence. In the segmentation result of α-expansion method for frame 19, the label assigned to pixels belonging to the left hand (gloves) of the boy has changed suddenly. Temporal consistency metric should penalise segmentation results for this change. In our segmentation results, label of pixels belonging to the left hand (gloves) of the boy remains same. Figure 6.9 shows the temporal consistency score for the *segment* representing the trouser. It is clear that for segmentation results of the α-expansion method, the minimum temporal consistency score is assigned to this *segment* at frame 19.

We have created synthetic videos to test our temporal consistency metric for presence of different scenarios in the segmentation results, such as occlusion, deocclusion, region splitting and region merging. Each row in figures 6.10 and 6.11 shows five frames from our synthetic video set. Our temporal consistency metric gives the best possible score of 1 the *segments* if

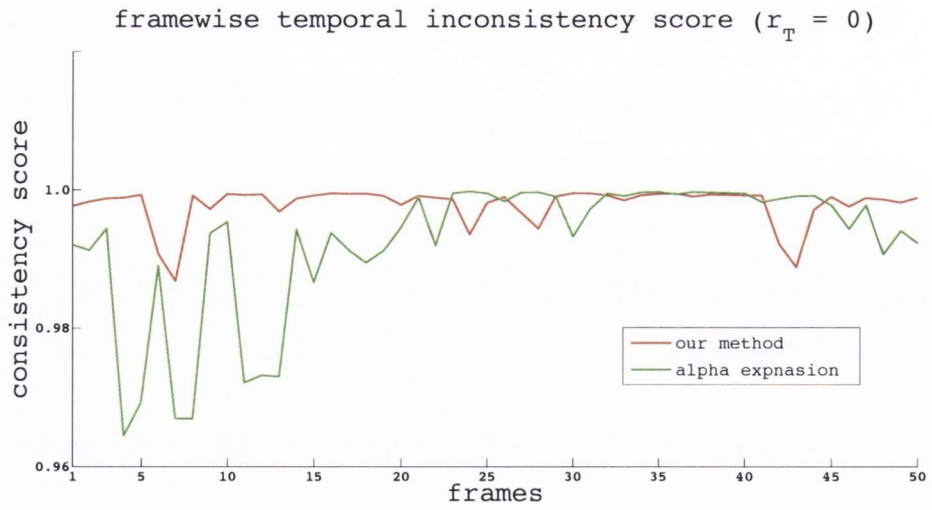


Figure 6.7: Temporal consistency scores for segmentation results of 50 frames of the *boy* video sequence using α-expansion method proposed by Boykov et al. [16] and our method. It is clear from the graph that temporal consistency of our segmentation results is considerably higher than the other method. For measuring the temporal inconsistency scores we have used block of five frames, quadratic curves with residual threshold equal to 0 and value of spatio temporal displacement constraint is set to 10. Two residual thresholds are selected to check the difference in the temporal consistency scores with ($r_t = 0.5$) or without ($r_t = 0$) image quantisation errors. Spatio temporal displacement constraint is set to 10 depending on the average *segment* size.

(a) Frames 17 – 21

(b) Label propagation results using the α-expansion method [16]. In segmentation result of central frame, label of pixel belonging to the left hand (left glove) of the boy is different from the previous frame. Label of *segment* representing the trouser is assigned to those pixels. Temporal consistency metric penalize the segmentation for this change (Figure 6.9).

(c) Our label propagation results. In segmentation results, label of pixels belonging to the left hand (left glove) of the boy remains same as previous frame.

Figure 6.8: Example of label change during label propagation using segmentation results of five sequential frames (frame numbers 17 – 21) of *boy* sequence.

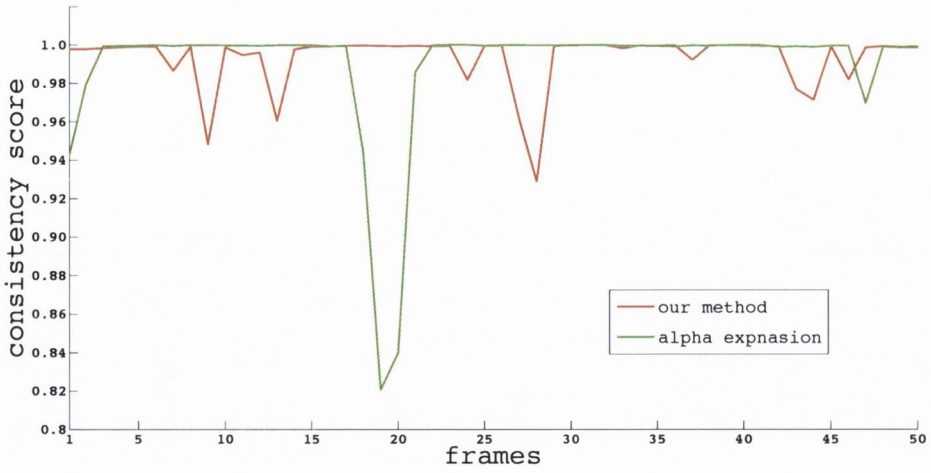
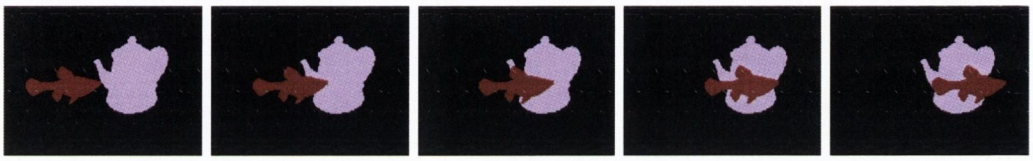


Figure 6.9: Temporal consistency score for *segment* representing the trouser in the *boy* video sequence. In segmentation results of α-expansion method, temporal consistency score is lowest at frame number 19. For measuring the temporal inconsistency scores we have used block of five frames, quadratic curves with residual threshold equal to 0 and value of spatio temporal displacement constraint is set to 10. In the vicinity of frame 19, a drop is observed in the α-expansion scores because the label assigned to pixels belonging to the left hand (gloves) of the boy has changed suddenly (Figure 6.8b). In our segmentation results, label of pixels belonging to the left hand (gloves) of the boy remains same.

any change in the shape and the position of *segments* is consistent over the whole block of frames. Changes which are not consistent over a block of frames are penalised. Figure 6.11 shows scenarios when changes in shape and position of *segment* are not consistent over the block of frames. Table 6.2 shows computed temporal consistency scores for these *segments* and frames.

All three of our metrics return scores in range of $[0, 1]$ where 0 is the worst and 1 is the best score. Scores of spatial accuracy metrics: spatial consistency and size consistency metrics indicate accuracy of the segmentation results with respect to the ground truth data. On the hand, scores of temporal consistency indicate changes in the segmentation results over the time. To evaluate segmentation results of a frame all three metrics are required. Higher scores from all three means higher accuracy and higher temporal consistency.

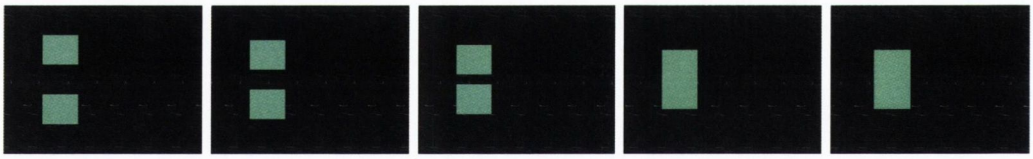
(a) No change



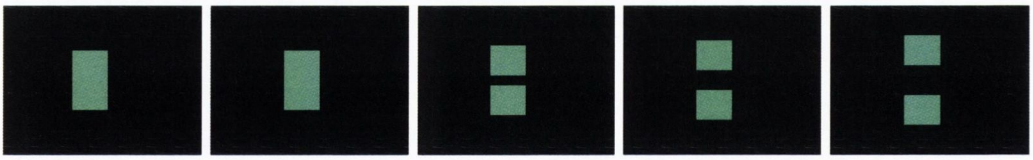
(b) Occlusion

(c) Appearance and disappearance of objects

(d) Motion



(e) Region merge



(f) Region split

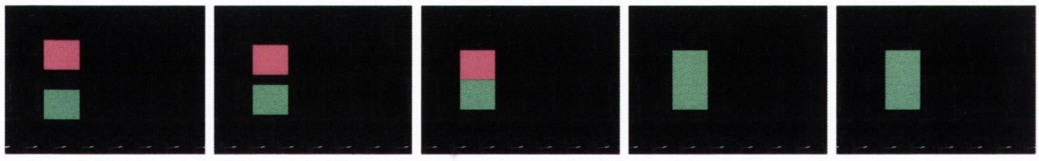
Figure 6.10: Temporal consistency metric examples with maximum consistency scores. In cases, when *segments* are (a) static (b) or occluding (c) or appearing and disappearing (d) or changing position (e) or merging (f) or splitting due to the consistent motion, temporal consistency score of segments and frames using our metric is 1. In these examples, block size is set to five frames per block, spatio temporal displacement constraint to 8 pixels and quadratic curves with residual threshold equal to 0 are used to fit points in *path*.

(a) Deformation

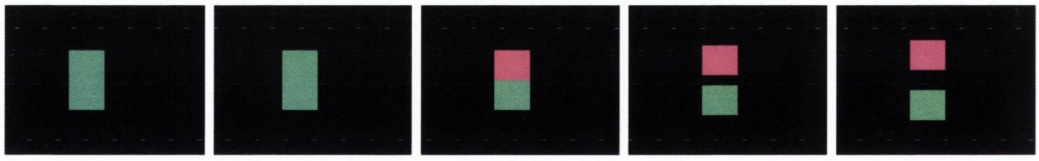
(b) Label change

(c) Label change

(d) Inconsistent motion



(e) Region merge and disappearance



(f) Region split and disappearance

Figure 6.11: Temporal inconsistency metric examples when temporal consistency of segments suffer due to inconsistent changes. In these examples, block size is set to five frames per block, spatio temporal displacement constraint to 8 pixels and quadratic curves with residual threshold equal to 0 are used to fit points in *path*. Table 6.2 shows temporal consistency score of these segments and frames.

row	frame	segments score								frame score
										
1	1	1	1							1
	2	1	1							1
	3	0.9979	0.9979							0.9979
	4	1	1							1
	5	1	1							1
2	1	1	0							0.7588
	2	1	0							0.7588
	3	1		0						0.7588
	4	1	0							0.7588
	5	1	0							0.7588
3	1	1	0							0.7588
	2	1	0							0.7588
	3	1		0						0.7588
	4	1		0						0.7588
	5	1		1						1
4	1	1	1		1	1				1
	2	1	1		1	1				1
	3	0.9989	1		0.9966	1				0.9989
	4	1	1		1	1				1
	5	1	1		1	1				1
5	1	1					1	1	1	1
	2	1					1	0	1	0.9640
	3	1					1	0	1	0.9640
	4	1					1		1	1
	5	1					1		1	1
6	1	1						1	1	1
	2	1						1	1	1
	3	1					1	0		0.9640
	4	1					1	0		0.9640
	5	1					1	1		1

Table 6.2: Temporal consistency scores of segments and frames shown in figure 6.11.

6.5 Conclusion

This chapter describes one of our major contributions, temporal consistency metric with two other metrics: spatial and size consistency metric. These metrics were able to detect and penalise spatial inaccuracies and temporal inconsistencies present in the segmentation results. Our spatial and size consistency metrics required ground truth for measuring the spatial accuracy of the video sequences. Temporal consistency metric did not require ground truth but successfully measures the consistency of the segmentation results over a block of frames.

In the next chapter, we present the results of our thesis and comparison of our results with state of the art methods.

Chapter 7

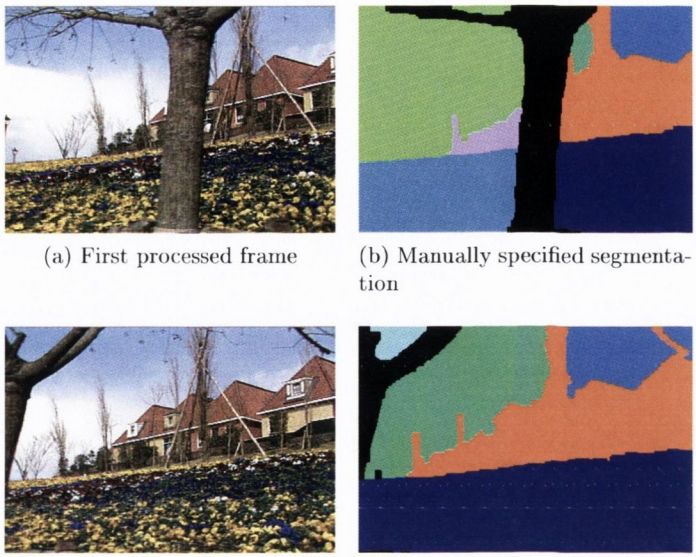
Results

The developed label propagation methods have been tested on eight video sequences (see Table 7.1). The frame size in each video was reduced to $240px \times 352px$. We compare our segmentation results with the α-expansion proposed by Boykov et al. [16] both subjectively and objectively. Spatial, size and temporal consistency metrics are used to compare the accuracy and the temporal consistency of the segmentation results.

For an objective analysis, evaluation metrics require segmentation results in the form of labelled masks. Wang and Collomosse's [1] results are excluded from the evaluation (objective) because their results are not available in a labelled form, but are instead available as video. Our label propagation reduces the number of comparisons between labels and pixels by limiting the search scope using the spatio-temporal displacement constraint (see section 3.3). Therefore, segmentation results may drift away from the original segmentation results over the time. To test the presence of drift, we compared the results of segmentation achieved by processing the video in the forward direction with those achieved by processing the video in the (temporally) reverse direction. In the latter, label propagation starts from the segmentation results of the frame which is processed last in the forward segmentation. Spatial and size consistency metrics are used to evaluate any change in the segmentation results achieved in both directions. It may be noted that the results achieved from forward processing are used as ground truth.

Our spatial, size and temporal metrics evaluate accuracy and temporal consistency of the segmentation results. To check any bias in the developed metrics, metric scores are compared with two state of the art evaluation metrics: scores of the spatial metric are compared with the Probabilistic Rand Index [129] and those of the temporal metric are compared with the

tqm metric [63].



(a) First processed frame

(b) Manually specified segmentation

(c) Last processed frame

(d) Segmentation result

Figure 7.1: The first and last processed frames of the *flower and garden* sequence in the forward direction and the corresponding segmentation results. In the manually specified segmentation, the first frame is divided into eight *segments*. The segmentation of the last processed frame produces six *segments*. Each *segment* is highlighted in a different colour.

Label propagation starts with the learning of colour and texture models for each *segment* present in the manually specified segmentation of the first processed frame. A Gaussian Mixture Model is used for colour modelling and a Normalised Texton Histogram is used for texture modelling. Figure 7.1, shows the first and last processed frames of the *flower and garden* sequence in the forward direction and the corresponding segmentation results. In manually specified segmentation, the first frame is divided into eight *segments*. Segmentation result of the last processed frame produces five *segments*. Figure 7.2 shows label propagation results of α-expansion (rows 2 and 6), results in the forward (rows 3 and 7) and reverse (rows 4 and 8) directions. It is observed that two of these *segments*, the flower bed to the left of the tree and bed to the right of the tree, have similar colour and texture properties. Consequently, the α-expansion method assigns a single label to both the *segments* representing the flower bed in following frames. However, with the spatio-temporal displacement constraint, our label propagation algorithm is able to propagate the labels present in the learned label set, including labels of the aforementioned similar segments in both directions.

In the reverse direction, after occluding bushes (between the houses and the tree) start appearing to the left of the tree, the label of the tree is assigned to a few pixels belonging

to the bushes to the left of the tree because colour and texture properties of the tree are similar to those of the bushes. If the colour and texture properties of pixels belonging to two neighbouring *segments* are similar, our label propagation method may misclassify some or all pixels from these *segments*. Except for this instance, segmentation results achieved in both directions by our method are very similar.

Figures 7.3 and 7.4 show spatial, size and temporal consistency scores for the segmentation results using the developed spatial, size temporal consistency metrics. We compare these with Probabilistic Rand Index [129] and *tqm* [63] metrics respectively. Spatial and size consistency metrics are used to measure the drift between our segmentation results achieved in the forward and the reverse direction. Our spatial consistency metric is similar to the Probabilistic Rand Index. However, it penalises any dissimilarities more strongly (Figure 7.3) than does Probabilistic Rand Index. Temporal consistency scores from our temporal consistency and *tqm* follow same trend. Both metrics give higher score to our label propagation results in comparison to the α-expansion method (Figure 7.4). It is observed from the figures that the developed method performed better in terms of accuracy and temporal consistency of the segmentation results when compared to α-expansion method and our segmentation results are similar in both directions.

In α-expansion method, any label from the *learned* label set can be assigned to any pixel. There is no influence of the segmentation results of the previous frame on the segmentation of the current frame. In other words, segmentation results of two consecutive frames could be very dissimilar. The temporal consistency metric penalises drastic changes in segmentation between consecutive frames. This can be observed in the graphs of temporal consistency metric scores of all test video sequences in figures 7.4, 7.8, 7.16, 7.12, 7.20, 7.24, 7.28 and 7.32. It is observed that the temporal consistency scores of the α-expansion has a higher variance as compared to results from the developed methods.

(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

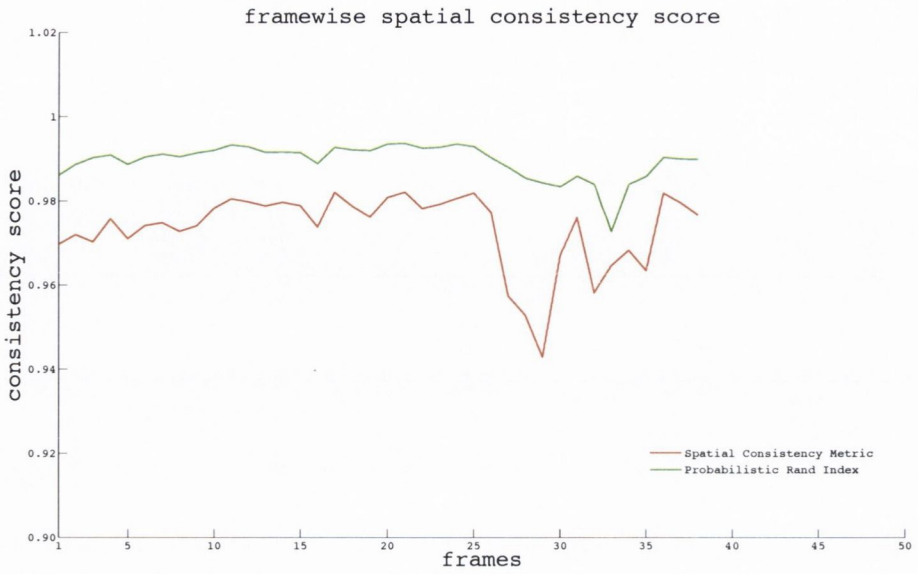
(e) Frames

(f) Label propagation results using the α-expansion method [16] in the forward direction.

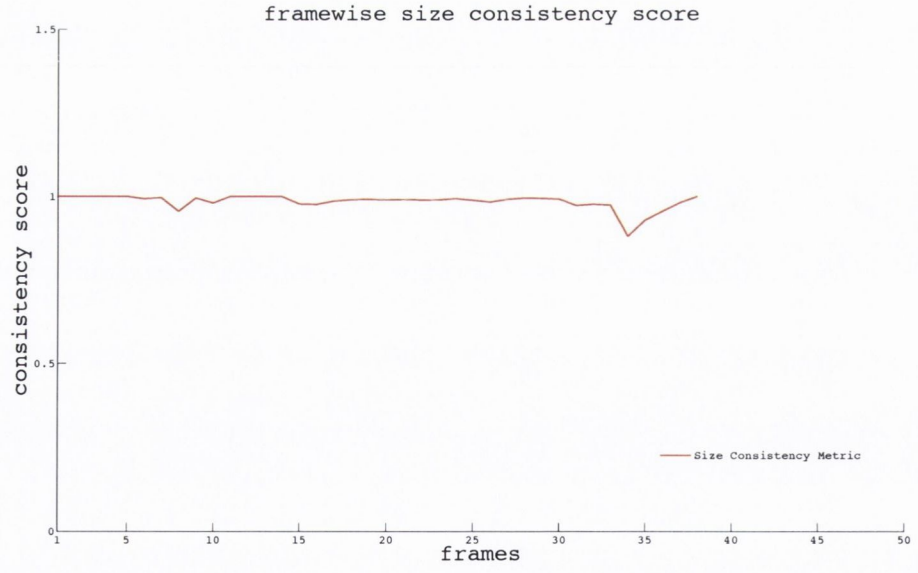
(g) Our label propagation results in the forward direction.

(h) Our label propagation results in the reverse direction.

Figure 7.2: Label propagation results for ten frames of the *flower and garden* sequence. Frames are selected at a regular interval of five frames.

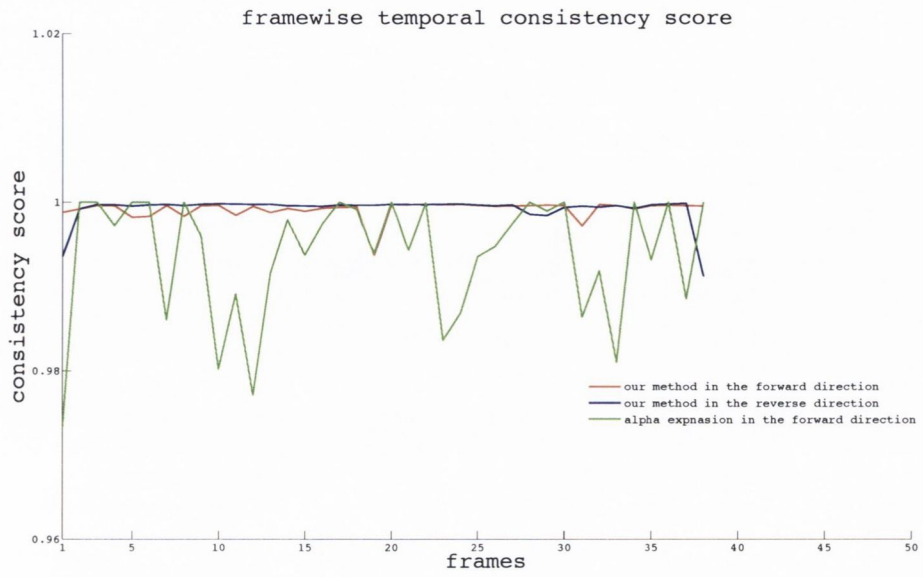


(a) Measuring drift in segmentation results of the forward and the reverse directions using our spatial consistency metric and Probabilistic Rand Index [129]. In the vicinity of frame 30, a drop is observed because in the reverse direction, a portion of flower bed starts appearing from the left hand side of the tree (third frame in Figure 7.2h). Pixels belonging to the flower bed are misclassified as the tree because the area of the new region is less than the error threshold. The score improves after a new label is assigned to the pixels belonging to this region.

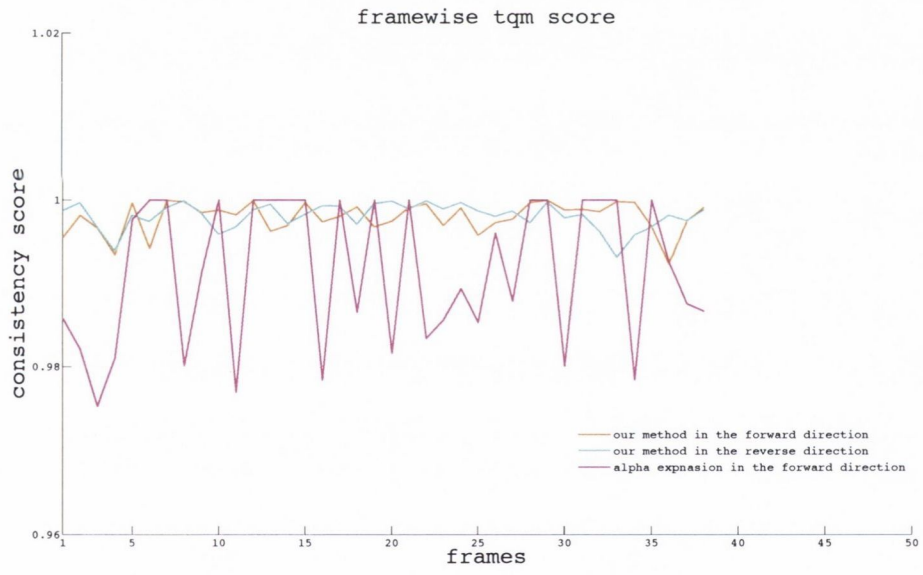


(b) Size consistency metric. A higher score is indicative of the fact that shapes of the *segments* remain similar in both directions.

Figure 7.3: Spatial and size consistency metrics scores for 40 frames of the *flower and garden* sequence. Note that the segmentation results in the forward direction are used as ground truth.

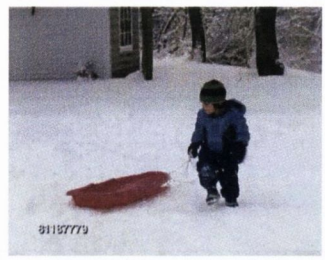


(a) Temporal consistency metric



(b) *tqm*

Figure 7.4: Temporal consistency measure using the developed temporal consistency metric and using *tqm* [63] for segmentation results of α-expansion [16] and the developed label propagation method in both the forward and the reverse directions for 40 frames of the *flower and garden* sequence. In α-expansion, labels of pixels belonging to the physical objects are not consistent. For instance, the label assigned to pixels belonging to the sky changes randomly over the time (figure 7.2b). As a result, both, the temporal consistency metric and *tqm* penalise α-expansion more when compared to our results. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.



(a) First processed frame

(b) Manually specified segmentation

(c) Last processed frame

(d) Segmentation result

Figure 7.5: The first and last processed frames of the *boy* sequence in the forward direction and the corresponding segmentation results. In manually specified segmentation, the first and last frames produce twelve and eleven *segments* respectively. Each *segment* is marked in a different colour.

Figure 7.5 shows the first and last processed frames of the *boy* sequence [1] in the forward direction and the corresponding segmentation results. Figure 7.6 shows label propagation results of α-expansion (rows 2 and 6) and our results in the forward (rows 3 and 7) and reverse (rows 4 and 8) directions. As in the previous case, the developed method performed better in terms of accuracy and temporal consistency of the segmentation results when compared to the α-expansion method, and segmentation results are similar in both directions (forward and reverse) (see Figures 7.7 and 7.8).

It may be noted in figure 7.5, that the colour and texture properties of the *segment* representing the wall, the bushes and the trees are similar. Consequently, α-expansion changes the labels assigned to pixels belonging to these *segments* frequently over the time. Our label propagation also misclassifies pixels belonging to one of these *segments* in a few frames (highlighted in orange colour figure 7.5b) in both directions.

(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

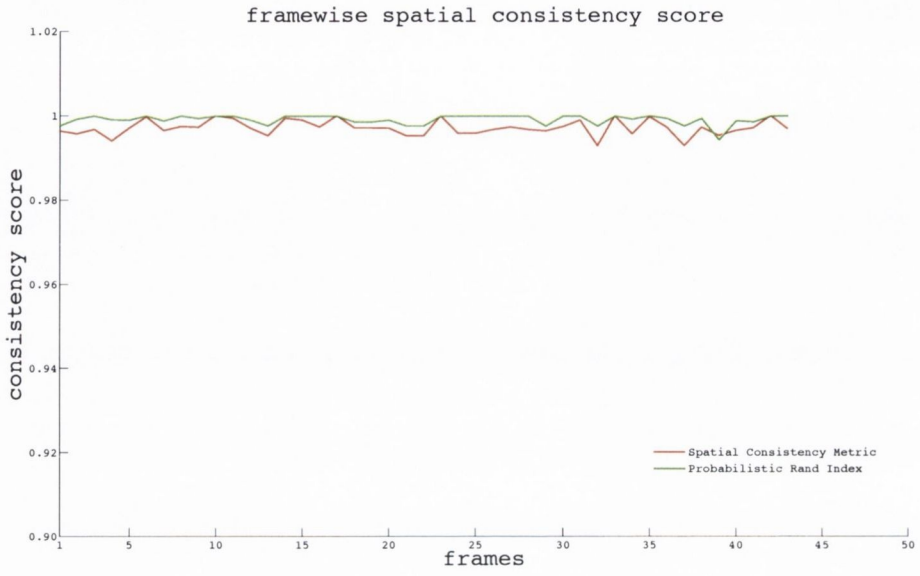
(e) Frames

(f) Label propagation results using the α-expansion method [16] in the forward direction.

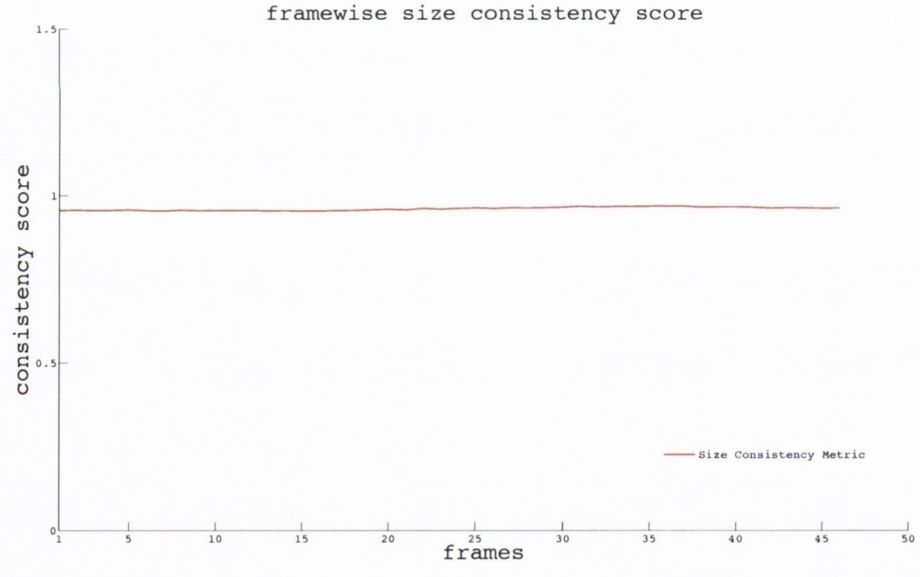
(g) Our label propagation results in the forward direction.

(h) Our label propagation results in the reverse direction.

Figure 7.6: Label propagation results for ten frames of the *boy* video sequence [1]. Frames are selected at a regular interval of twenty frames.

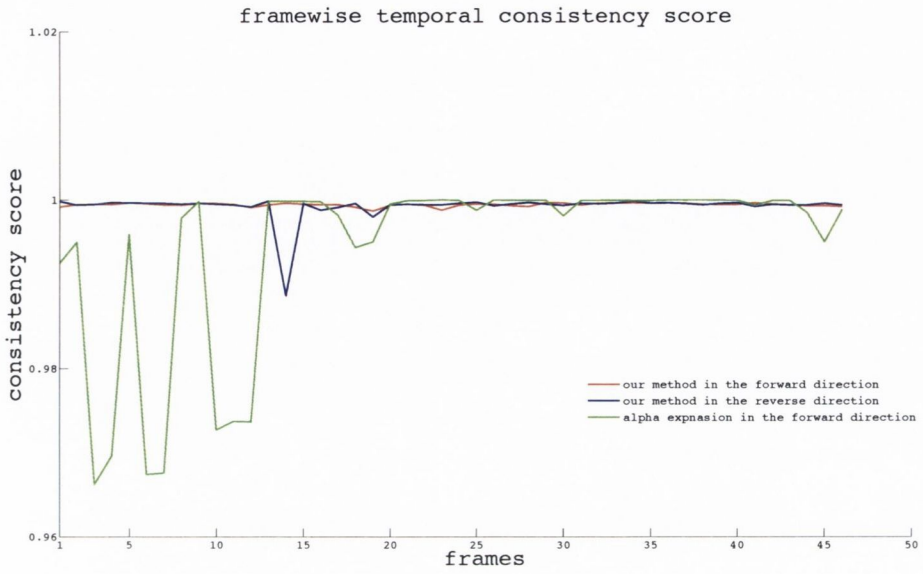


(a) Measuring drift between segmentation results of the forward and reverse directions using the developed spatial consistency metric and Probabilistic Rand Index [129]

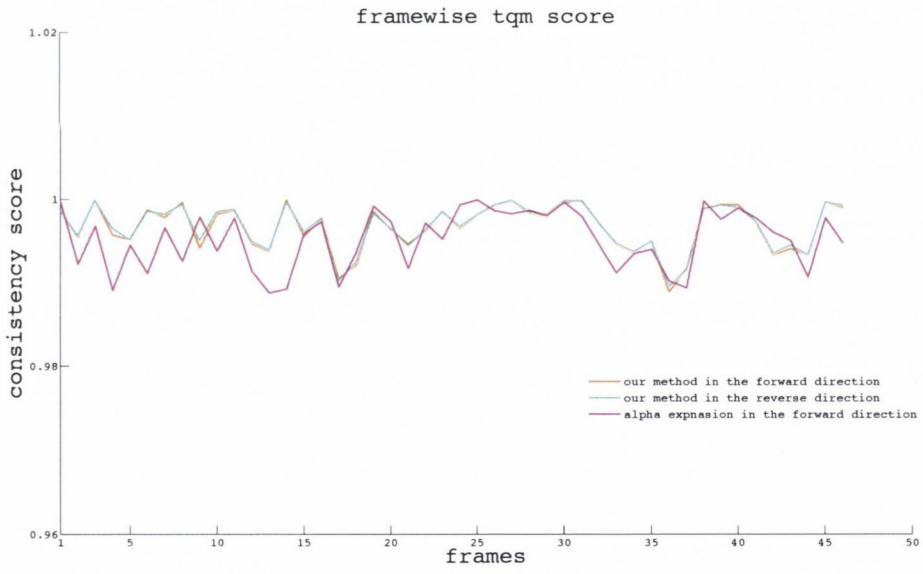


(b) Size consistency metric

Figure 7.7: Spatial and size consistency metrics scores for fifty frames of the *boy* video sequence. Segmentation results in the forward direction are used as ground truth.



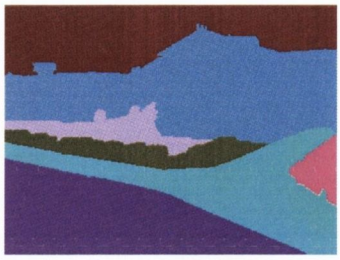
(a) Temporal consistency metric



(b) *tqm*

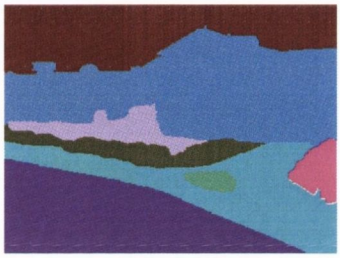
Figure 7.8: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and our method in both the forward and reverse directions for fifty frames of the *boy* sequence. Inconsistencies in α-expansion results are due to frequent change in pixels labelled as the wall, bushes and trees in the video sequence. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.

(a) First processed frame



(b) Manually specified segmentation

(c) Last processed frame



(d) Segmentation result

Figure 7.9: The first and last processed frames of the *PETS 2001* video sequence in the forward direction and the corresponding segmentation results. In the specified segmentation results, the first and last frames produce seven and eight *segments* respectively. Each *segment* is highlighted in a different colour.

Figure 7.9 shows the first and last processed frames of a *PETS 2001* video sequence in the forward direction and the corresponding segmentation results. Figure 7.10 shows label propagation results of α-expansion (rows 2 and 6), our results in the forward (rows 3 and 7) and reverse (rows 4 and 8) directions for subjective analysis. In the video sequence, a new object, *car* appears after few frames. *car* is missing from the manually specified segmentation of the first frame. Consequently, α-expansion assigns label of the most similar label class to the pixels belonging to the *car*. However, a new label is assigned to pixels belonging to the *car* in our segmentation results.

It can be seen in figures 7.11 and 7.12 that the developed method performed better in terms of accuracy and temporal consistency of the segmentation results as compared to α-expansion method and our segmentation results are similar in both directions. In α-expansion, labels assigned to pixels belonging to physical objects are not consistent. For instance, some pixels belonging to the buildings often get mis-classified as parked cars (Figures 7.10b and 7.10f). As a result, the temporal consistency metric and *tqm* penalise α-expansion results more in comparison to our results.

(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

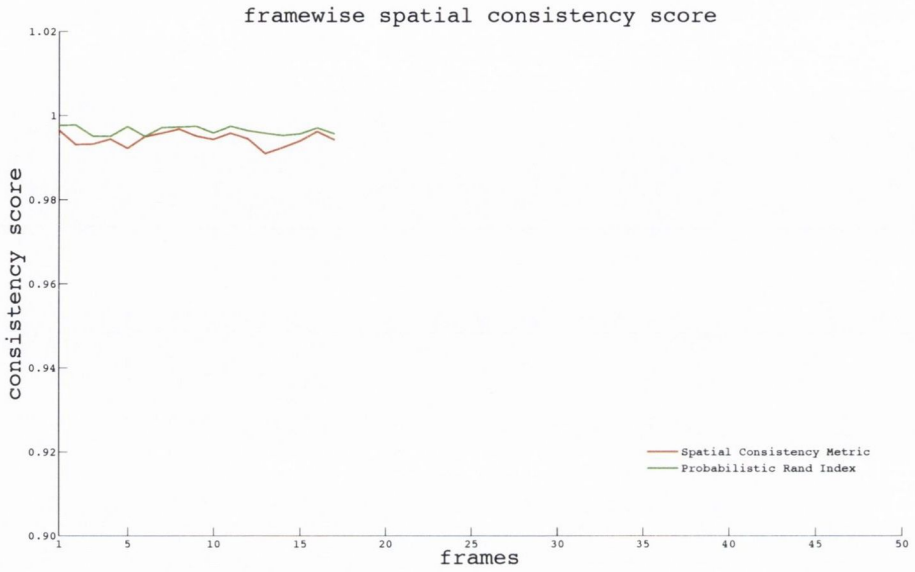
(e) Frames

(f) Label propagation results using the α-expansion method [16] in the forward direction.

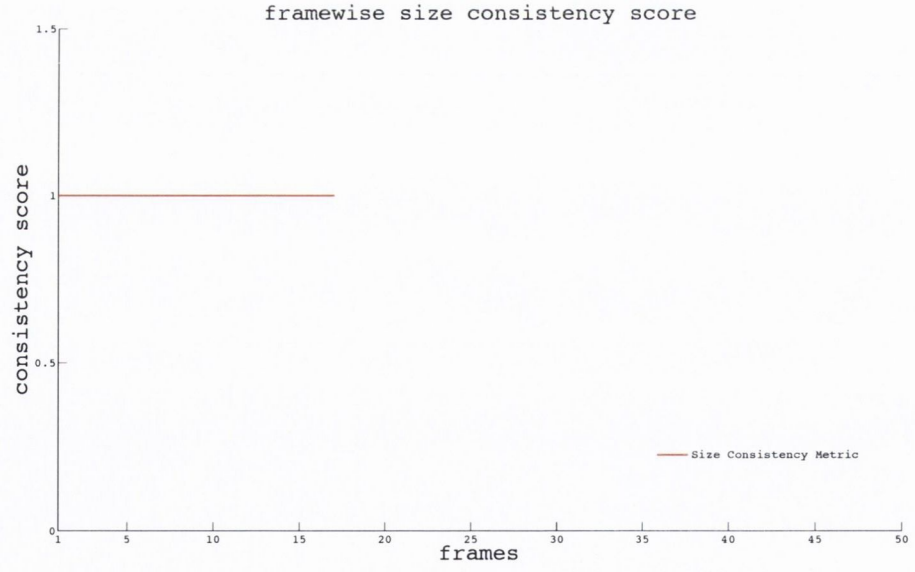
(g) Our label propagation results in the forward direction.

(h) Our label propagation results in the reverse direction.

Figure 7.10: Label propagation results for ten frames of the *PETS 2001* video sequence. Frames are selected at a regular interval of two frames.

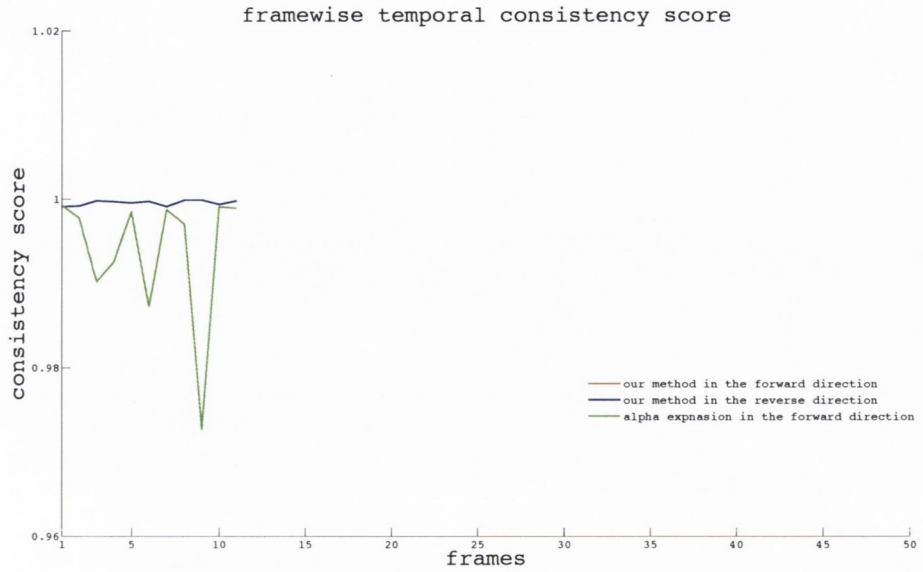


(a) Measuring drift in between segmentation results of the forward and the reverse directions using our spatial consistency metric and Probabilistic Rand Index [129]. A higher score is indicative of the fact that segmentation results are similar in both directions. In other words, the drift in segmentation results is negligible.

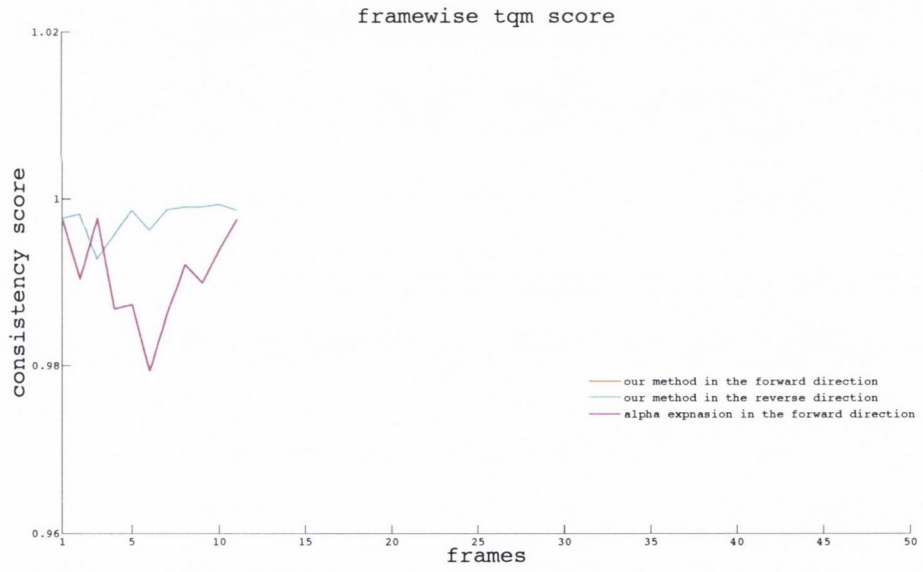


(b) Size consistency metric. A higher score is indicative of the fact that shapes of the *segments* remain similar in both directions.

Figure 7.11: Spatial and size consistency metrics scores for 15 frames of a *PETS 2001* video sequence. Segmentation results in the forward direction are used as ground truth.

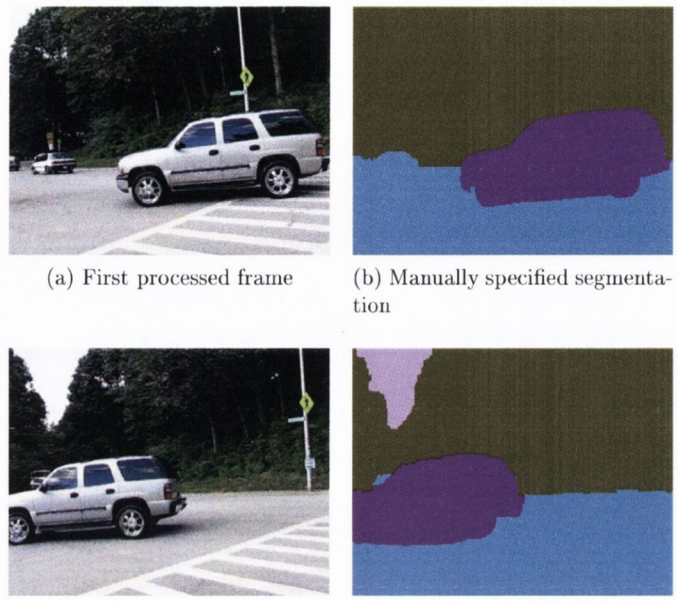


(a) Temporal consistency metric



(b) *tqm*

Figure 7.12: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and the developed method in both the forward and reverse directions for 15 frames of a *PETS 2001* video sequence. In α-expansion, labels of pixels assigned to physical objects are not consistent. For instance, some pixels belonging to the buildings often get mis-classified as parked cars (figures 7.10b and 7.10f). Consequently, the temporal consistency metric and *tqm* penalise α-expansion results more in comparison to our results. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.



(a) First processed frame

(b) Manually specified segmentation

(c) Last processed frame

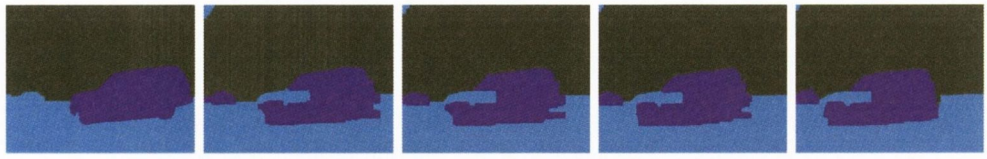
(d) Segmentation result

Figure 7.13: The first and last processed frames of the *car1* sequence in the forward direction and the corresponding segmentation results. In the specified segmentation results, the first and last frames are divided into three and four *segments* respectively. Each *segment* is highlighted in a different colour.

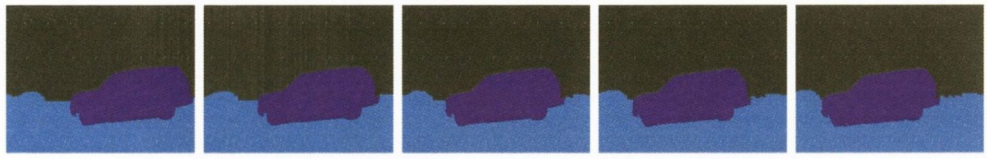
Figure 7.13 shows the first and last processed frames of the *car1* sequence in the forward direction and the corresponding segmentation results. In the *car1* video sequence a (left to right) moving car is captured by a (right to left) moving camera. Figure 7.14 shows label propagation results of α-expansion (rows 2 and 6), our results in the forward (rows 3 and 7) and reverse (rows 4 and 8) directions for subjective analysis. In the video sequence a new object, *sky* appears after few frames. *sky* is missing from the manually specified segmentation of the first frame therefore, α-expansion assigns label of the most similar *segment* (*car*) to the pixels belonging to the *sky*. However, a new label is assigned to the pixels belonging to the *sky* in our segmentation results.

It is observed both from subjective and objective (Figures 7.15 and 7.16) evaluation our developed method performed better in terms of accuracy and temporal consistency of the segmentation results when compared to α-expansion method. Our segmentation results are similar in both directions therefore, no drift is measured.

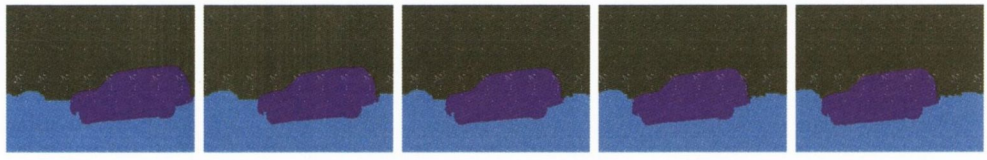
(a) Frames



(b) Label propagation results using the α-expansion method [16] in the forward direction.

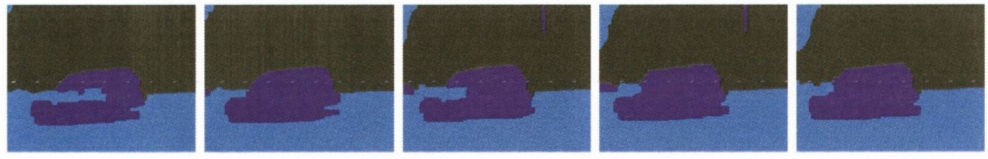


(c) Our label propagation results in the forward direction.

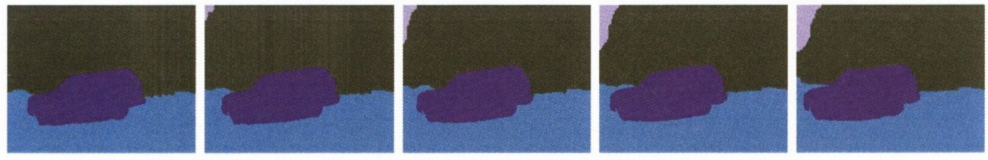


(d) Our label propagation results in the reverse direction.

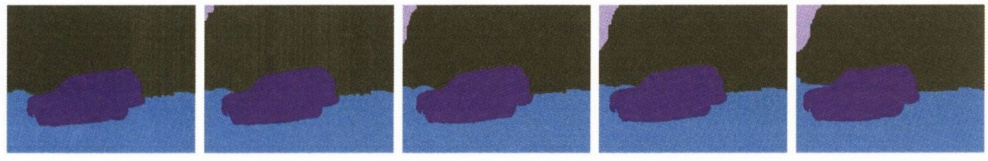
(e) Frames



(f) Label propagation results using the α-expansion method [16] in the forward direction.

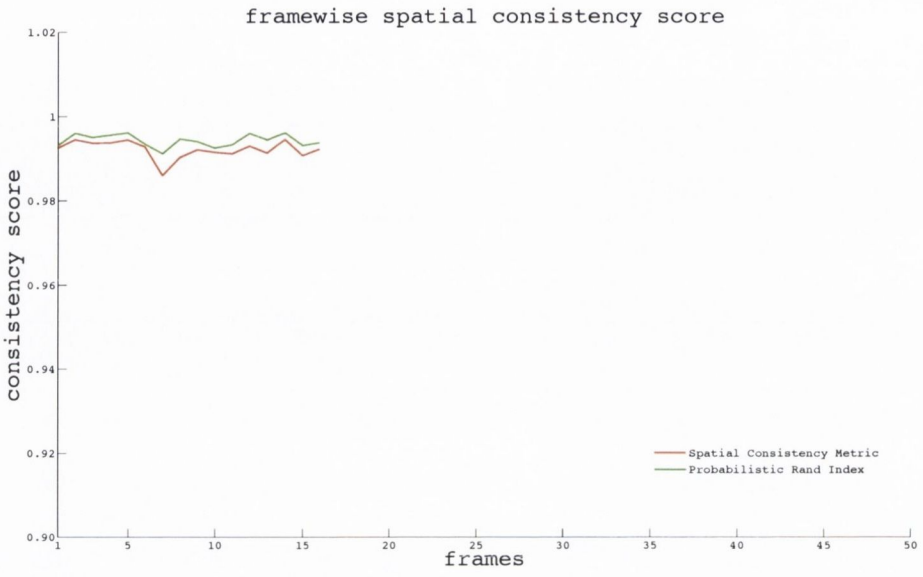


(g) Our label propagation results in the forward direction.

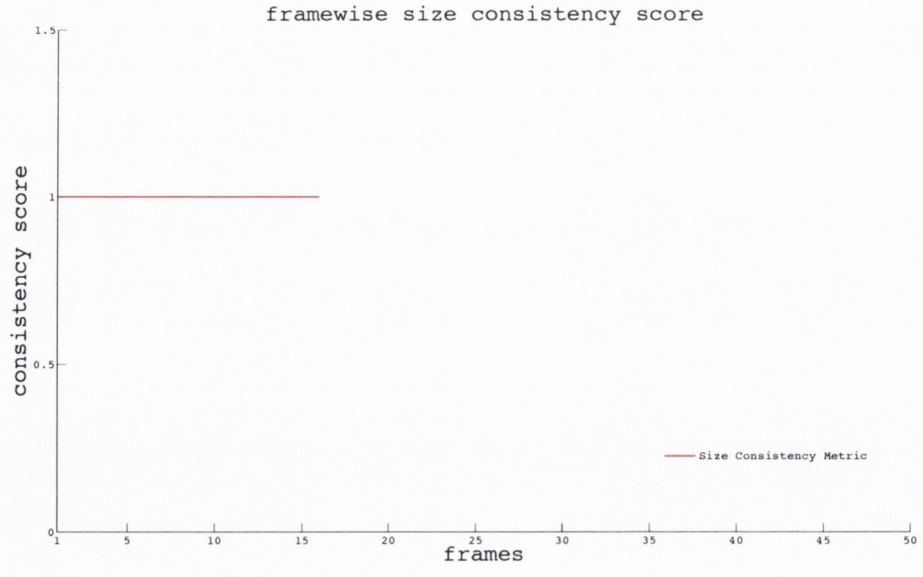


(h) Our label propagation results in the reverse direction.

Figure 7.14: Label propagation results for ten frames of the *car1* sequence. Frames are selected at regular interval of two frames.

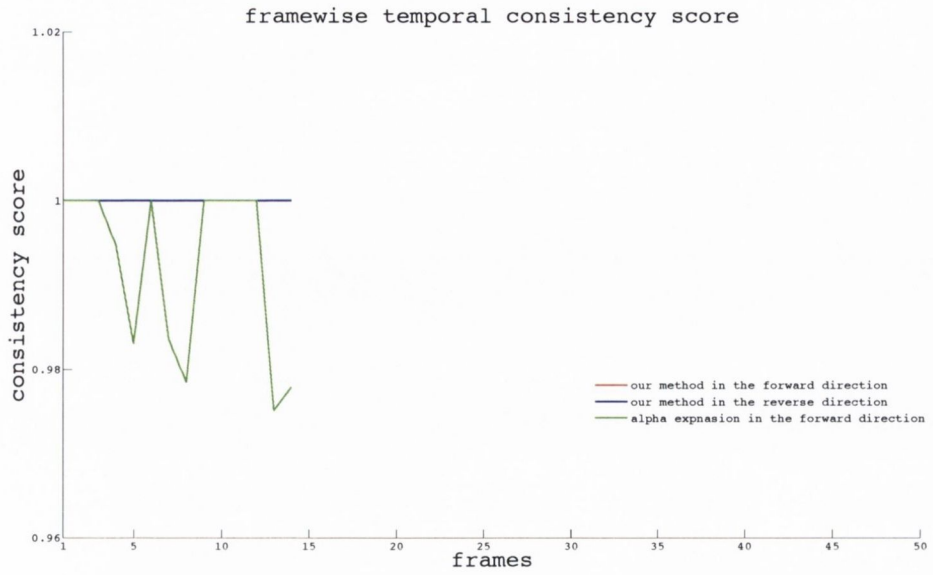


(a) Measuring drift in between segmentation results of the forward and the reverse directions using our spatial consistency metric and Probabilistic Rand Index [129]. Our segmentation results are similar in both directions: drift is not significant enough to be observable.

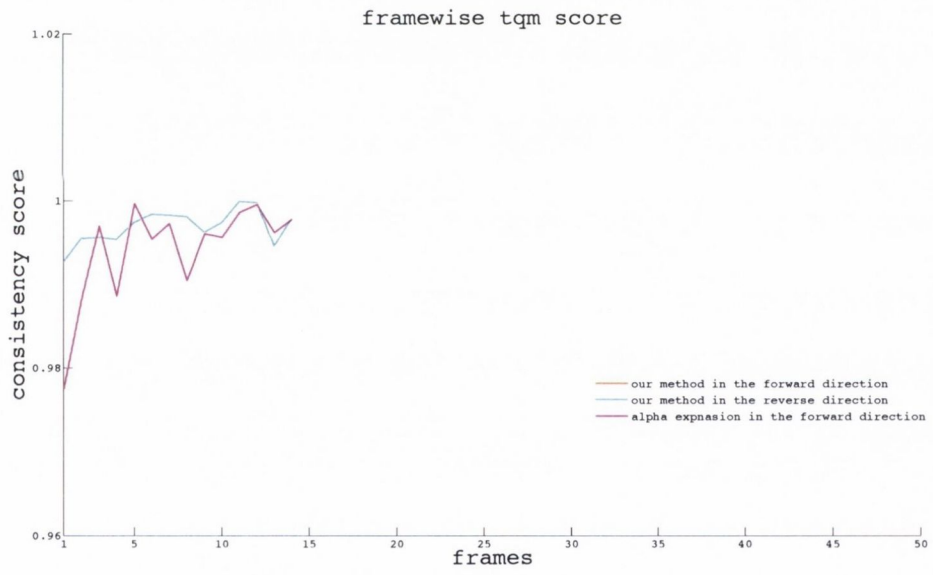


(b) Size consistency metric. A higher score is indicative of the fact that shapes of the *segments* remain similar in both directions.

Figure 7.15: Spatial and size consistency metrics scores for 19 frames of the *car1* video sequence. Segmentation results in the forward direction are used as ground truth.



(a) Temporal consistency metric



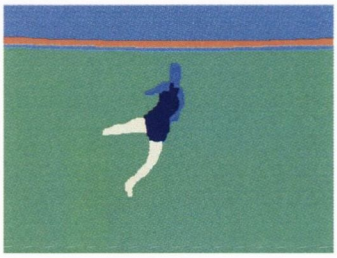
(b) *tqm*

Figure 7.16: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and the developed method in both the forward and the reverse directions for nineteen frames of the *car1* sequence. It can be seen in the figure 7.13 that the colour and texture properties of several pixels near the front of the car and the road are similar. Consequently, during label propagation α-expansion assigns either label to such pixels (figures 7.14b and 7.14e). The developed label propagation method is able to handle this issue with the help of *shape* and motion cues. A higher temporal consistency score is assigned to our segmentation results. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.

(a) First processed frame

(b) Manually specified segmentation

(c) Last processed frame



(d) Segmentation result

Figure 7.17: The first and the last processed frames of the *yunakim* sequence in the forward direction and the corresponding segmentation results. In the specified segmentation results, the first and the last frames are divided into nine and seven *segments* respectively. Each *segment* is highlighted in a different colour.

Figure 7.17 shows the first and last processed frames of the *yunakim* sequence [1] in the forward direction and the corresponding segmentation results. In the *yunakim* video sequence a moving ice skater is captured by a moving camera. Figure 7.18 shows label propagation results of α-expansion (rows 2 and 6), our results in the forward (rows 3 and 7) and in the reverse (rows 4 and 8) directions. In this video sequence, ice skater is rotating respective to the camera and her face and hair are occluded and deoccluded several times in the video sequence. In the forward direction, the developed method is able to assign correct labels to the pixels belonging to the face and hair. The sizes of the *segments* representing the face and hair is lesser than those of other *segments*. In the reverse direction, some pixels belonging to the face and hair are misclassified after deocclusion. Moreover in the forward direction, a single label is assigned to all signboards in the segmentation results of the last frame. In the reverse direction, our algorithm learns the *segments*' properties using the segmentation results of the last processed frame. Consequently, a single label is assigned to all signboards in the reverse direction in comparison to the forward direction (rows $3, 4, 7$ and 8). The size consistency metric penalises the segmentation results for this error (figure 7.20).

(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

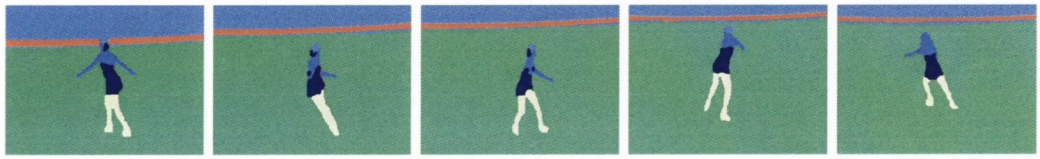
(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

(e) Frames

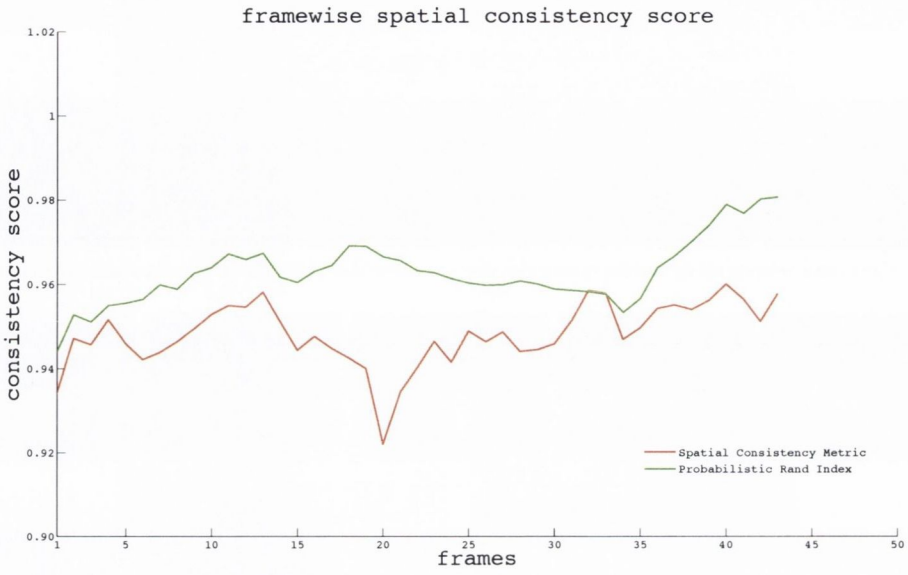
(f) Label propagation results using the α-expansion method [16] in the forward direction.

(g) Our label propagation results in the forward direction.

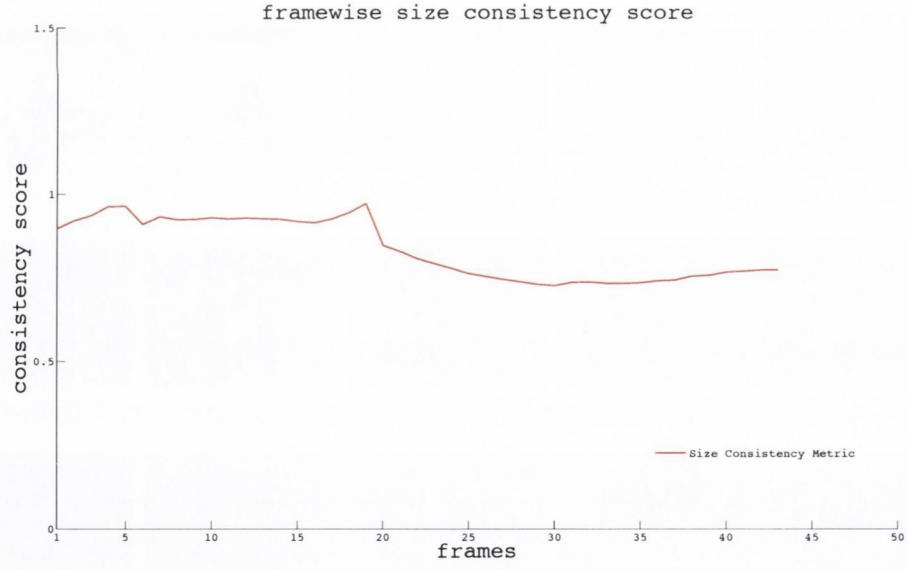


(h) Our label propagation results in the reverse direction.

Figure 7.18: Label propagation results for ten frames of the *yunakim* video sequence [1]. Frames are selected at a regular interval of seven frames.

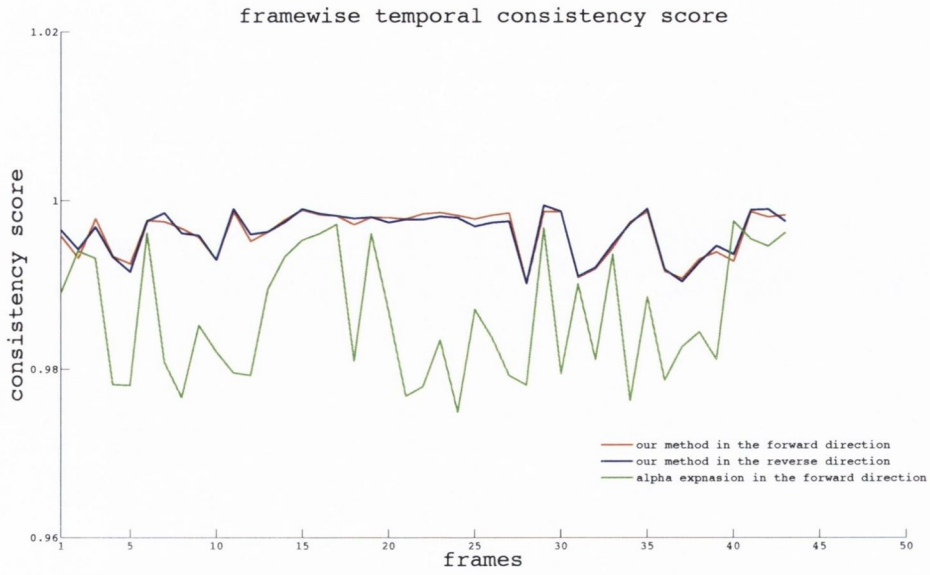


(a) Measuring drift in between segmentation results of the forward and the reverse directions using the developed spatial consistency metric and Probabilistic Rand Index [129].

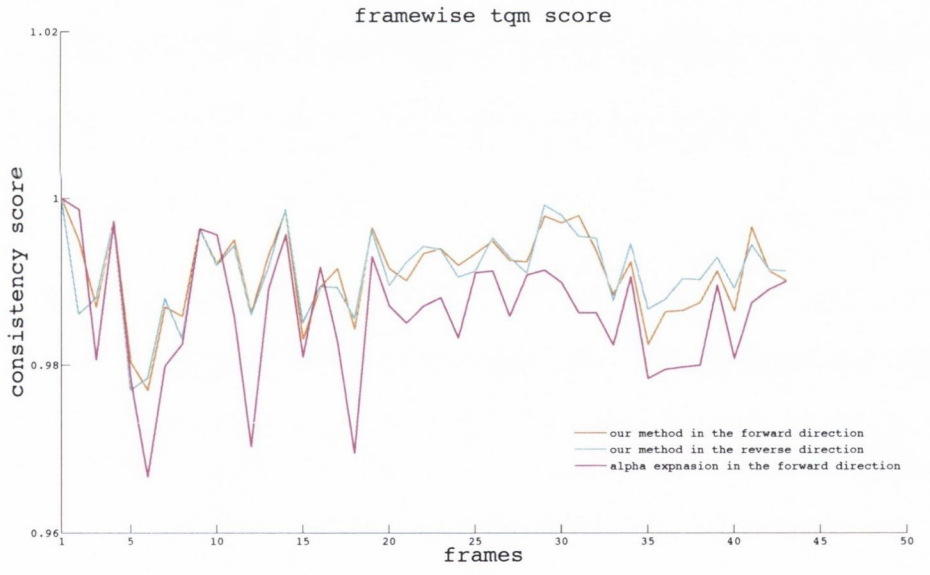


(b) Size consistency metric

Figure 7.19: Spatial and size consistency metrics scores for 45 frames of the *yunakim* video sequence. Segmentation results in the forward direction are used as ground truth. Spatial and size consistency scores mainly suffers due to the misclassification of the pixels belonging to the small *segments*, such as *segments*, representing the face and hair during the label propagation in the reverse direction.



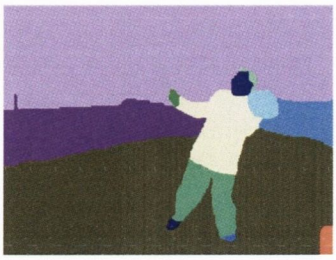
(a) Temporal consistency metric



(b) *tqm*

Figure 7.20: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and the developed method in both the forward and reverse directions for 45 frames of the *yunakim* sequence. It can be seen in figure 7.17 that the colour and texture properties of several pixels of the floor and one of the signboard, and pixels of the hands and legs are similar. As a result, α-expansion assigns either label to such pixels (figures 7.18b and 7.18e) during label propagation. Both, the developed temporal consistency metric, and *tqm* penalise α-expansion results for any change in the labels assigned to pixels belonging to these *segments*. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.

(a) First processed frame



(b) Manually specified segmentation

(c) Last processed frame

(d) Segmentation result

Figure 7.21: The first and the last processed frames of the *dance* sequence in the forward direction and the corresponding segmentation results. In the specified segmentation results, the first and last frames are divided into thirteen and ten *segments* respectively. Each *segment* is highlighted in a different colour.

Figure 7.21, shows the first and the last processed frames of the *dance* sequence [1] in the forward direction and the corresponding segmentation results. In this video sequence, a person is moving, specifically dancing, with respect to a static camera, and the hands and boots of the dancer occluding and deoccluding several times in the video sequence. In the forward direction, our method is able to assign correct labels to the pixels belonging to the hands and boots. However, our method misclassifies some of the pixels belonging to the hands and the boots after deoclussion in the reverse direction (figure 7.22). A few pixels of two *segments*, the terrain to the left of the dancer and the terrain to the right of the dancer, have similar colour and texture properties (Figure 7.21). As a result, α-expansion misclassifies some of the pixels of belonging to these *segments*. However, with the spatio-temporal displacement constraint, our label propagation method is able to assign correct labels to these pixels in both directions.

It is observed both from subjective and objective evaluation, from figures 7.23 and 7.24, that accuracy and temporal consistency of results is significantly improved in comparison to the α-expansion method and our segmentation results are similar in both directions.

(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

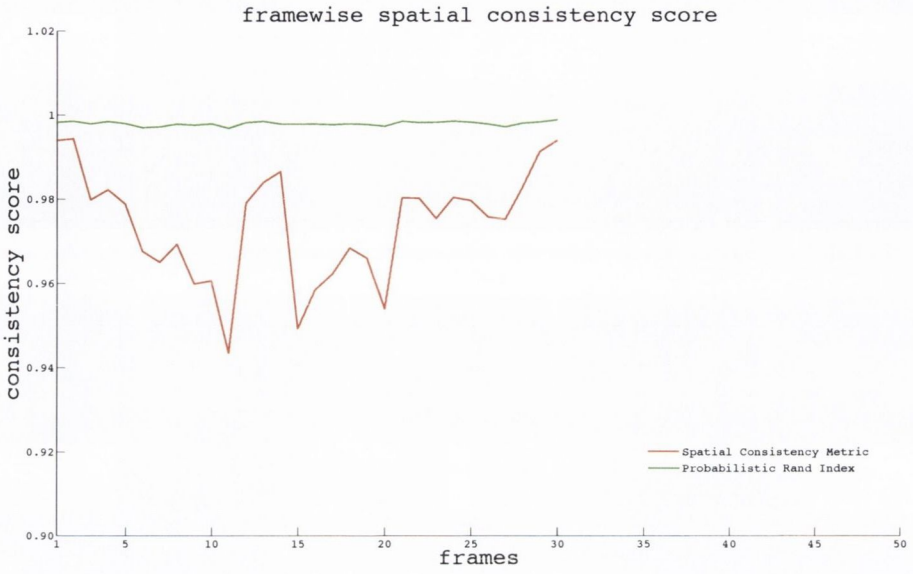
(e) Frames

(f) Label propagation results using the α-expansion method [16] in the forward direction.

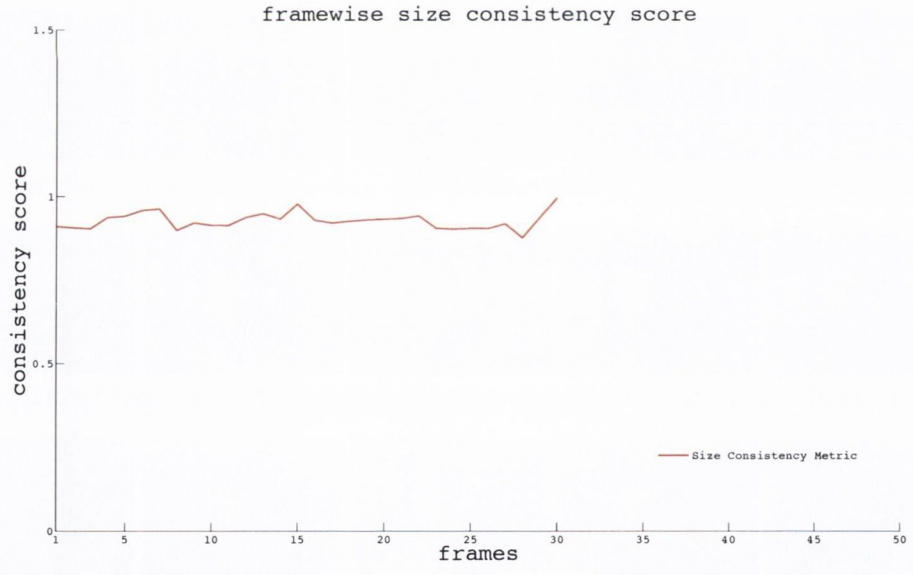
(g) Our label propagation results in the forward direction.

(h) Our label propagation results in the reverse direction.

Figure 7.22: Label propagation results for ten frames of the *dance* sequence. Frames are selected at regular interval of three frames.

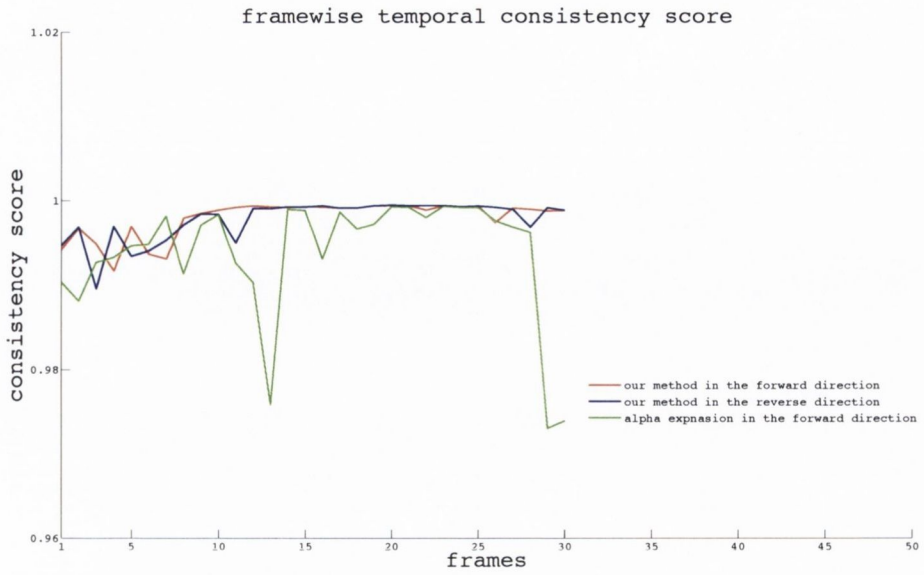


(a) Measuring drift in between segmentation results of the forward and the reverse directions using our spatial consistency metric and Probabilistic Rand Index [129].

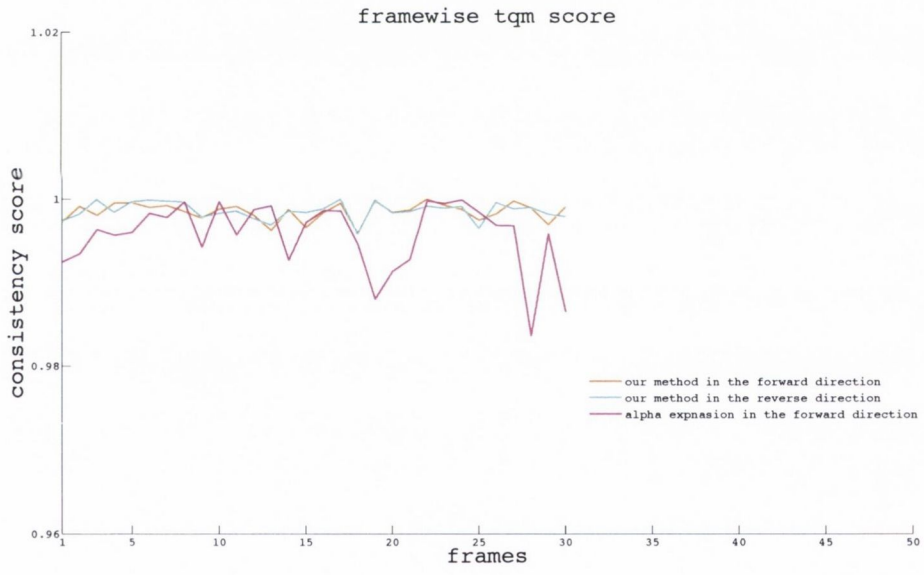


(b) Size consistency metric

Figure 7.23: Spatial and size consistency metrics scores for 33 frames of the *dance* video sequence. Segmentation results in the forward direction are used as ground truth. Our label propagation results are penalised due to misclassification of the pixels belonging to the small *segments* like the hands and the boots in the reverse direction.



(a) our temporal consistency metric

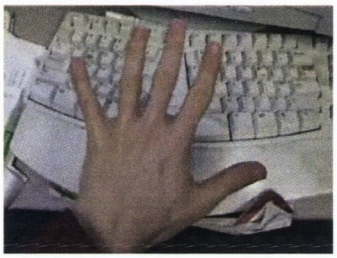


(b) tqm

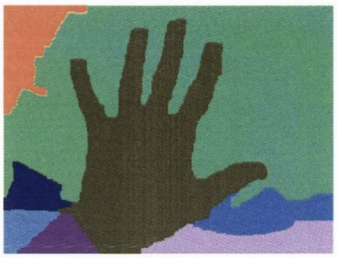
Figure 7.24: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and the developed method in both the forward and the reverse directions for 33 frames of the *dance* sequence. It can be seen in the figure 7.21 that both the colour and texture properties of several pixels of the terrain to the left of the dancer and the terrain to the right of the dancer are similar. Consequently, α-expansion assigns either label to such pixels (figures 7.22b and 7.22e) during the label propagation. Both our temporal consistency metric and *tqm* are penalising α-expansion results for any change in the labels of pixels belonging to these *segments*. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.

(a) First processed frame

(b) Manually specified segmentation



(c) Last processed frame



(d) Segmentation result

Figure 7.25: The first and the last processed frames of the *hands2* sequence in the forward direction and the corresponding segmentation results. In both the specified segmentation results, the first and the last frames are divided into eight *segments*. Pixels belonging to each *segment* are highlighted in different colours.

Figure 7.25 shows the first and last processed frames of the *hands2* sequence [2] in the forward direction and the corresponding segmentation results. In the *hands* video sequence a (left to right) moving hand is captured by a static camera. Figure 7.26 shows label propagation results of α-expansion (rows 2 and 6), our results in the forward (rows 3 and 7) and reverse (rows 4 and 8) directions for subjective analysis. It is observed both from subjective and objective evaluation, from figures 7.27 and 7.28, that accuracy and temporal consistency of results is significantly improved in comparison to the α-expansion method and our segmentation results are similar in both directions.

(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

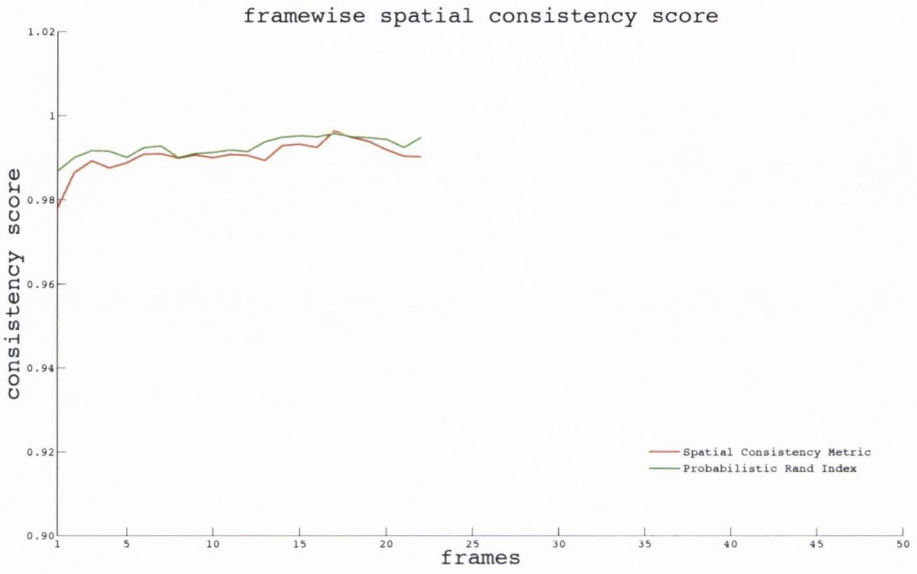
(e) Frames

(f) Label propagation results using the α-expansion method [16] in the forward direction.

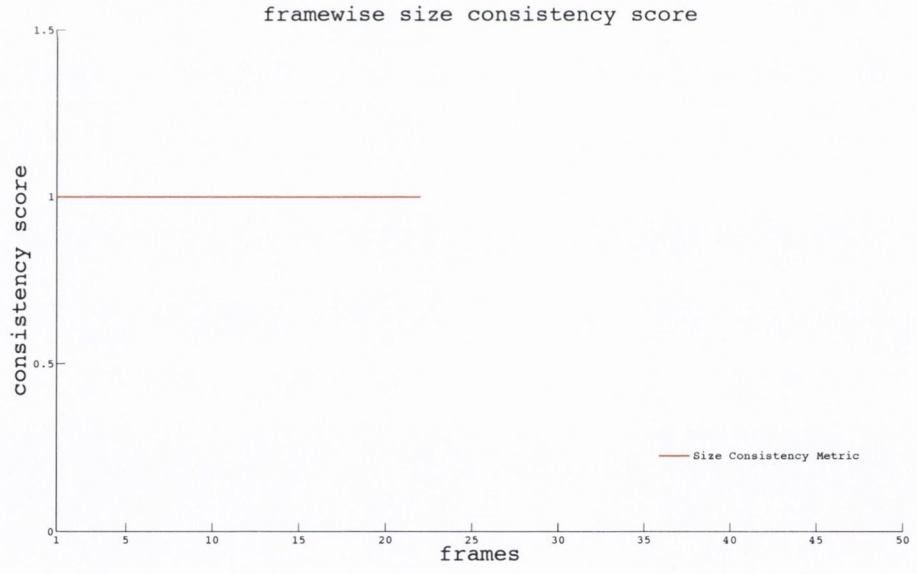
(g) Our label propagation results in the forward direction.

(h) Our label propagation results in the reverse direction.

Figure 7.26: Label propagation results for ten frames of the *hands2* sequence [2]. Frames are selected at a regular interval of three frames.

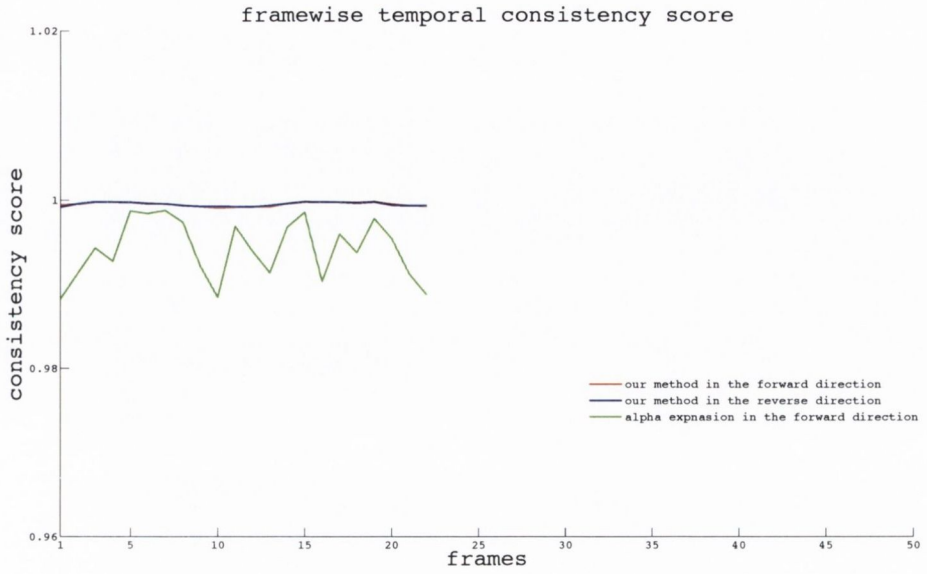


(a) Measuring drift between segmentation results of the forward and the reverse directions using the developed spatial consistency metric and Probabilistic Rand Index [129].

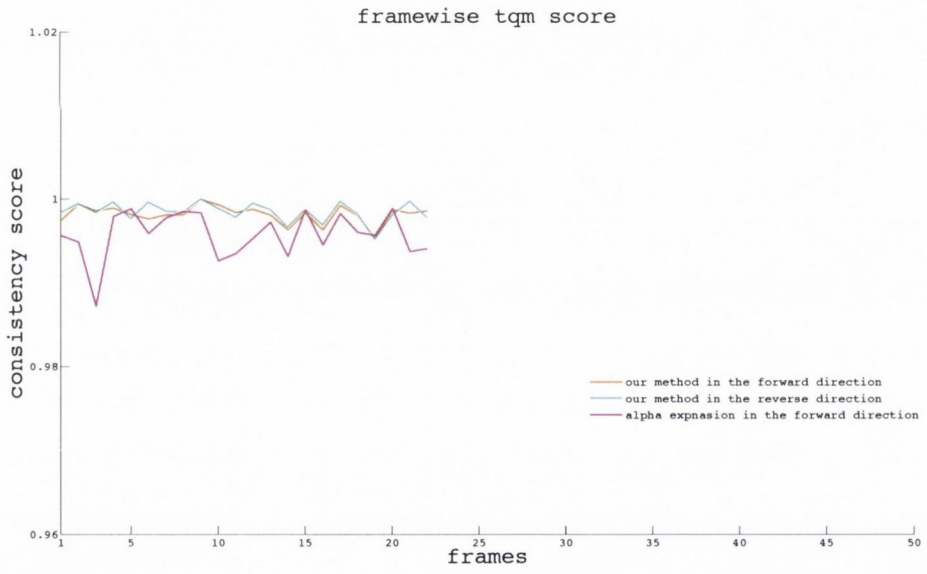


(b) Size consistency metric. A higher score is indicative of the fact that shapes of the *segments* remain similar in both directions.

Figure 7.27: Spatial and size consistency metrics scores for 25 frames of the *hands2* video sequence. Segmentation results in the forward direction are used as ground truth.

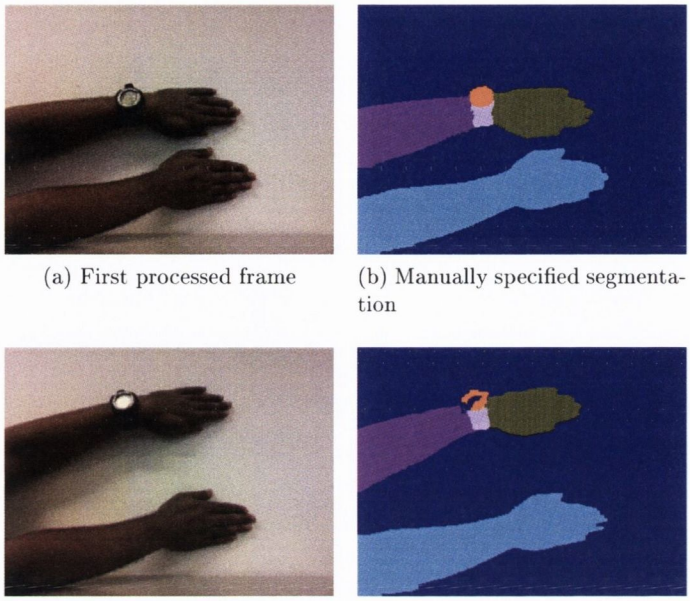


(a) The developed temporal consistency metric



(b) *tqm*

Figure 7.28: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and the developed method in both the forward and the reverse directions for 25 frames of the *hands2* sequence. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.



(a) First processed frame

(b) Manually specified segmentation

(c) Last processed frame

(d) Segmentation result

Figure 7.29: The first and last processed frames of the *hands* sequence in the forward direction and the corresponding segmentation results. In both the specified segmentation results, the first and the last frames are divided into six *segments*. Each *segment* is highlighted in a different colour.

Figure 7.29 shows the first and the last processed frames of the *hand* sequence in the forward direction and the corresponding segmentation results. In the *hand* video sequence a (bottom to top) moving hand is captured by a moving camera. Figure 7.30 shows label propagation results of α-expansion (rows 2 and 6), our results in the forward (rows 3 and 7) and reverse (rows 4 and 8) directions for subjective analysis. A few pixels of three *segments*, representing the hands and fingers, have similar colour and texture properties. Consequently, α-expansion method assigns the same label to pixels belonging to *segments* representing hands in subsequent frames. However, with the spatio-temporal displacement constraint, our label propagation algorithm is able to propagate the labels present in the learned label set including labels of the *similar* segments in both directions. Some of the pixels belonging to the watch are also misclassified in all three results: α-expansion and our results in the forward and reverse directions due to the changes in lighting.

It is observed both from subjective and objective evaluation, from figures 7.31 and 7.32, that accuracy and temporal consistency of results is significantly improved in comparison to the α-expansion method and our segmentation results are similar in both directions.

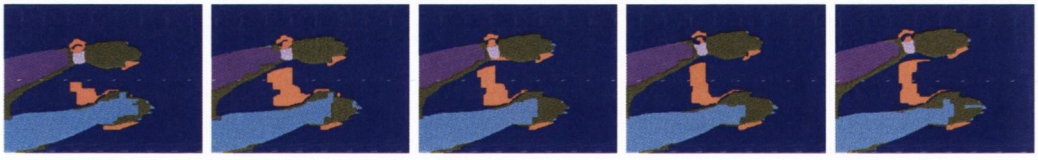
(a) Frames

(b) Label propagation results using the α-expansion method [16] in the forward direction.

(c) Our label propagation results in the forward direction.

(d) Our label propagation results in the reverse direction.

(e) Frames

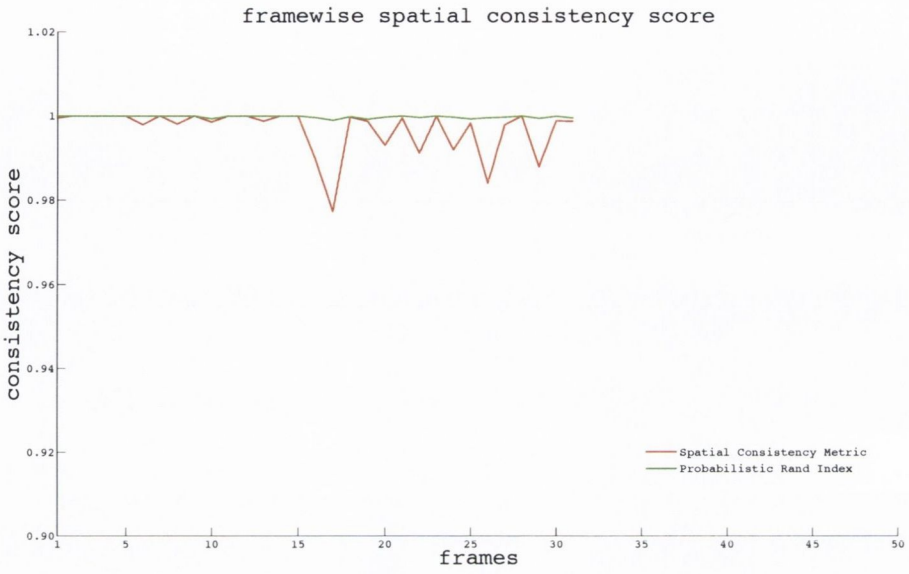


(f) Label propagation results using the α-expansion method [16] in the forward direction.

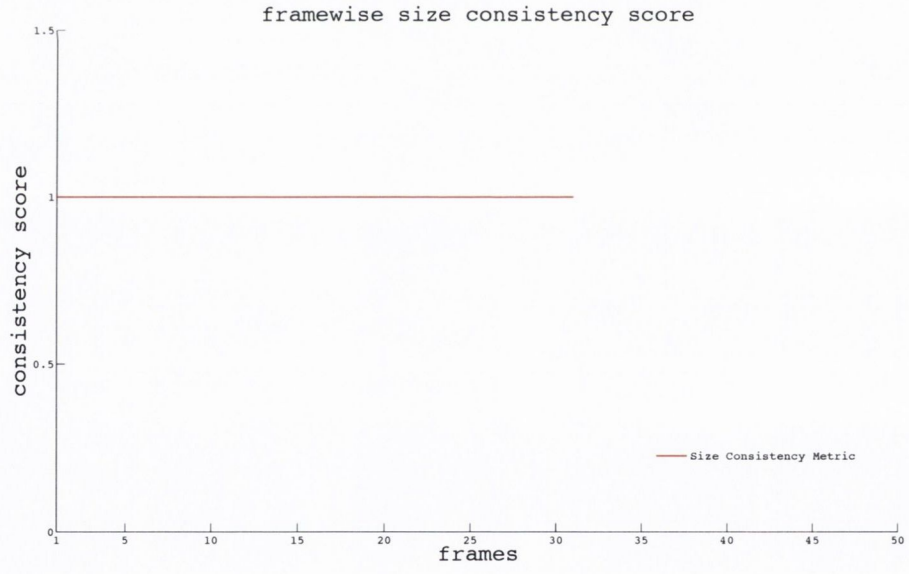
(g) Our label propagation results in the forward direction.

(h) Our label propagation results in the reverse direction.

Figure 7.30: Label propagation results for ten frames of the *hands* sequence. Frames are selected at regular interval of three frames.

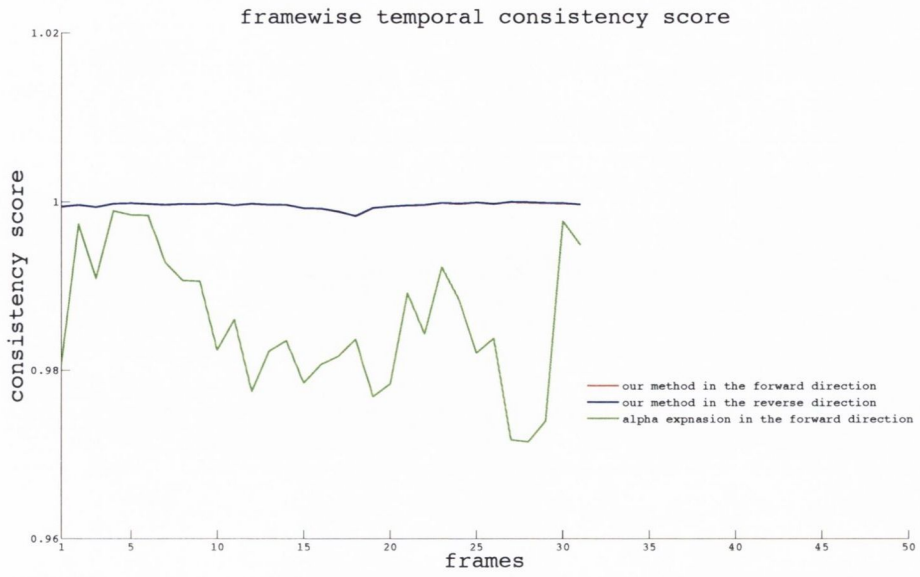


(a) Measuring drift in between segmentation results of the forward and the reverse directions using our spatial consistency metric and Probabilistic Rand Index [129].

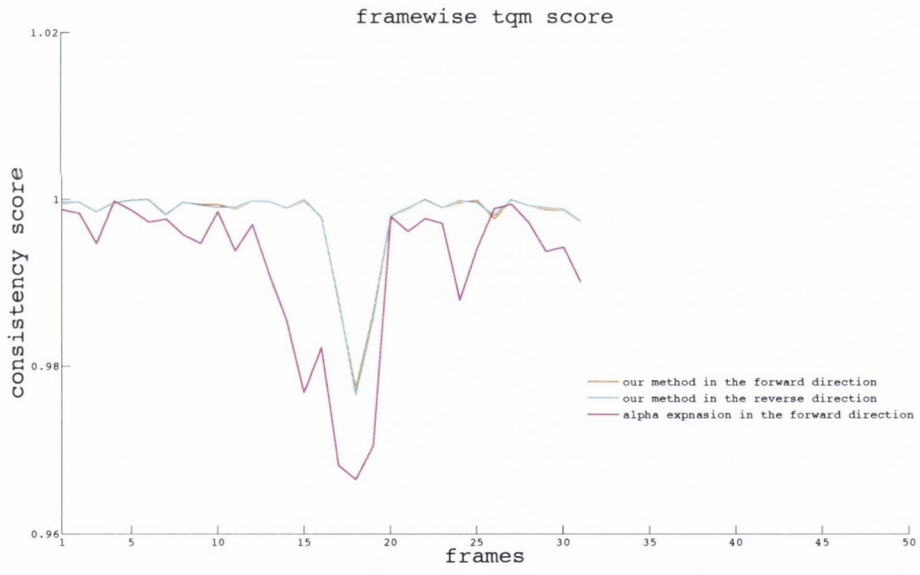


(b) Size consistency metric. A higher score is indicative of the fact that shapes of the *segments* remain similar in both directions.

Figure 7.31: Spatial and size consistency metrics scores for 35 frames of the *hands* video sequence. Segmentation results in the forward direction are used as ground truth.



(a) The temporal consistency metric



(b) *tqm*

Figure 7.32: Temporal consistency measure using the developed metric and *tqm* [63] for segmentation results of α-expansion [16] and the developed method in both the forward and the reverse directions for 35 frames of the *hands* sequence. It can be seen in figure 7.29 that both the colour and texture properties of several pixels of three *segments*, the two hands and fingers, are similar. Consequently, α-expansion assigns one of three label to such pixels (figures 7.30b and 7.30e) during label propagation. Both our temporal consistency metric and *tqm* penalise α-expansion results for any change in the labels of pixels belonging to these *segments*. For measuring the temporal consistency score our metric uses a block of five frames. It may be noted that we use quadratic curves with a residual threshold equal of zero and a spatio-temporal displacement constraint of ten.

No.	Video Sequence	No. of Frames	Metric	our method		α-expansion
				forward	reverse	
1.	*flower and garden*	40	Spatial	0.9735		
			PRI	0.9909		
			Size	0.9845		
			Temporal	0.9991	0.9992	0.9921
			tqm	0.9979	0.9980	0.9924
			time/frame	61 sec.	60 sec.	52 sec.
2.	*boy* [1]	50	Spatial	0.9971		
			PRI	0.9992		
			Size	0.9641		
			Temporal	0.9994	0.9986	0.9945
			tqm	0.9966	0.9967	0.9952
			time/frame	67 sec.	65 sec.	58 sec.
3.	*PETS 2001* [106]	15	Spatial	0.9955		
			PRI	0.9997		
			Size	1		
			Temporal	0.9996	0.9996	0.9938
			tqm	0.9976	0.9976	0.9908
			time/frame	63 sec.	63 sec.	52 sec.
4.	*car1* [107]	19	Spatial	0.9964		
			PRI	0.9984		
			Size	1		
			Temporal	0.9998	0.9998	0.9923
			tqm	0.9969	0.9969	0.9941
			time/frame	44 sec.	44 sec.	32 sec.
5.	*yunakim* [1]	45	Spatial	0.9481		
			PRI	0.9630		
			Size	0.8367		
			Temporal	0.9962	0.9961	0.9850
			tqm	0.9911	0.9911	0.9862
			time/frame	69 sec.	66 sec.	57 sec.
6.	*dance* [1]	33	Spatial	0.9740		
			PRI	0.9980		
			Size	0.9276		
			Temporal	0.9980	0.9978	0.9938
			tqm	0.9985	0.9986	0.9956
			time/frame	64 sec.	63 sec.	57 sec.
7.	*hand2* [2]	25	Spatial	0.9904		
			PRI	0.9927		
			Size	1		
			Temporal	0.9995	0.9995	0.9941
			tqm	0.9982	0.9982	0.9958
			time/frame	66 sec.	66 sec.	58 sec.
8.	*hands*	35	Spatial	0.9968		
			PRI	0.9998		
			Size	1		
			Temporal	0.9996	0.9996	0.9858
			tqm	0.9976	0.9976	0.9919
			time/frame	48 sec.	48 sec.	41 sec.

Table 7.1: Spatial and temporal consistency scores of test video sequences.

Table 7.1 shows scores of spatial consistency, size consistency, temporal consistency, Probabilistic Rand Index (PRI), *tqm* and time/frame (average time per frame in seconds) metrics for results of the α-expansion in the forward direction and results of our label propagation method in both the forward and the reverse directions for all the eight test videos.

7.1 Conclusion

This chapter presented our final label propagation results. Developed spatial, size and temporal consistency metrics were used to measure the quality of our segmentation results in comparison to the α-expansion method. It was shown that our label propagation methods outperformed the α-expansion method in terms of the temporal consistency. Our label propagation reduced the number of comparisons between labels and pixels by limiting the search scope using the spatio-temporal displacement constraint. Therefore, segmentation results may drift away from the original segmentation results over time. To test the presence of drift, spatial and size consistency metrics were used to compare the results of segmentation achieved by processing the video in the forward direction with those achieved by processing the video in the (temporally) reverse direction. It was shown that label propagation results in both the directions were similar except for the pixels belonging to the smaller physical objects.

To investigate any bias in the developed metrics for our label propagation methods, our label propagation results were also evaluated using two state of the art methods: Probabilistic Rand Index [129] and *tqm* [63]. Results from these metrics were compared with the results of our evaluation metrics. It was observed that evaluation scores from spatial consistency metric and temporal consistency metric followed trend similar to the evaluation scores of the Probabilistic Rand Index and *tqm* metrics respectively.

In the next chapter, we present the conclusion of our thesis and discuss directions for future work.

Chapter 8

Conclusions and Future Work

This chapter summarises the conclusions of this research and provides directions for future work on temporally consistent video segmentation and evaluation metrics.

8.1 Conclusions

This body of research addresses the problem of segmenting a video sequence in a temporally consistent fashion. It presents a novel method based on α-expansion based label propagation, to address *non-class based* video segmentation, appearance of new objects, and occlusion and deocclusion of regions.

Evaluation of temporal consistency in video is a problem which has not been well addressed in previous literature, primarily due to a need for pixel based ground truth for each frame of the video. We present a new metric for effectively evaluating the temporal consistency of video segmentation that does not require any ground truth.

Spatio temporal displacement constraint: Energy minimisation especially α-expansion based label propagation is a popular approach for temporally consistent video segmentation. In this thesis, we extend this framework. Our method allows α-expansion based methods to produce a best labelling with lesser pixel to label comparisons. Our method is also able to deal with *non-class based* video segmentation.

In a video sequence new objects can appear at any point in time. New objects may be an instance of a learned label class or an unknown class (not present in training data). α-expansion [16] may assign any label from the learned label set to any pixel of the current

frame. This is how it is able to handle the appearance of a new instance of any learned label class in the current frame. However, it will assign the same label to all instances of a physical object even though unique labels are assigned to different instances of the physical object in the manually specified segmentation of the first processed frame. Moreover, if new object belongs to a unknown class, α-expansion [16] will assign incorrect labels to the pixels belonging to it. In our approach, each pixel of the current frame is compared only with a small number of labels which are present in its spatio-temporal neighbourhood. As a result, our method is able to assign unique labels to all instances of a physical object. However, a pre-processing step is required for to accommodate the appearance of new objects of known and unknown label classes.

After label propagation, each *segment* should ideally represent a complete physical object or a part of a physical object. A *segment* should not have pixels belonging to more than one physical object. If the colour and texture of pixels belonging to two neighbouring *segments* are similar, label propagation methods such as α-expansion [16], Wang et al. [1] and our label propagation method mis-classified some or all of the pixels from these *segments*.

The imposed spatio temporal displacement constraint reduces the number of comparisons between labels and pixels by limiting the search scope. However, segmentation results may drift away from the original segmentation results over time. To quantify such drift, spatial and size consistency metrics were used to compare the results of segmentation achieved by processing the video in the forward direction with those achieved by processing the video in the (temporally) reverse direction. It was shown that label propagation results in both the directions were similar except for the pixels belonging to the smaller physical objects.

Handling appearance of new objects: In section 3.6, we develop an algorithm to accommodate for the appearance of new objects. Unlike the methods proposed by Wang et al. [6, 49] which compares label propagation results of the current frame and previous frames in a post-processing step, our algorithm uses the energy minimisation framework. Pixels belonging to new objects were identified successfully by introducing an extra label *void* in the learned set. In our algorithm there is no restriction on the number of new regions that can appear in the next frame. New regions can also appear from any location in the frame. However, to avoid segmentation errors, a new label is only assigned if the area of the detected new object is bigger than a certain size.

Our method for handling appearance of new object is a pre-processing step that adds an extra computational cost to our label propagation method. To reduce this cost colour cues were considered for data cost (texture cues are ignored). The developed method performs substantially better when the colour cues of pixels belonging to the neighbouring *segments* were different. However, it mis-classified pixels belonging to new objects, even though the other properties like texture and shape are different.

Improving label propagation results for rigid objects: Our label propagation method and α-expansion [16] may mis-classify pixels of neighbouring *segments* if their brightness, colour and texture properties are similar. In chapter 4, we discuss a post-processing method to improve label propagation results for rigid objects. The developed algorithm used motion and shape cues of the *segments* for this purpose. The cues are learned using the results from previous label propagation. It was observed that the algorithm is able to produce accurate and temporally consistent segmentation results for the video sequences even though the colour and texture cues of the pixels belonging to the neighbouring *segments* were similar.

The developed post-processing method for improving the label propagation results for rigid objects can be used to improve the label propagation results of any state of the art label propagation method like α-expansion [16]. However, as with the pre-processing step for handling the appearance of the new objects, this step also increased the overall computational cost of the approach.

Learning the physical explanations behind changes and occlusion boundaries: In chapter 5, we propose a method for learning the physical explanations behind changes and occlusion boundaries. We demonstrate that any change in shape of the *segments* in the current frame can be categorised as occlusion or deocclusion. Occlusion and deocclusion information of the *segments* was further used for identifying the occlusion boundaries. Occlusion and deocclusion information of the *segments* can help the higher level video processing, viz. scene understanding. Our method models changes in the label propagation results of any state of the art label propagation method like α-expansion [16] as occlusion and deocclusion.

The accuracy of our learned physical explanations - occlusion or deocclusion -depends on both the accuracy of segmentation results and the motion cues. In other words, errors present segmentation results and learned motion cues will cause incorrect physical explanations and

false occlusion boundaries.

Evaluation metrics: In chapter 6, we propose spatial, size and temporal consistency metrics for measuring the accuracy of images/frames segmentation and the temporal consistency of video segmentation. Moreover, we showed that our accuracy metrics are robust to under-segmentation in results.

In literature, there are no robust approaches for measuring temporal consistency without ground truth. The temporal consistency metric developed in this work measures temporal consistency of the segmentation results without using detailed ground truth. However, the time complexity of our temporal metric is very large and of the order of $O(M^{n-1})$ where M is the maximum number of the boundary pixels that may be present in the spatio-temporal neighbourhood of a pixel and n is number of frames in a block of frames.

Results in chapter 7 show that developed metrics can work for any state of the art label propagation method if segmentation results are in a specific format. To investigate any bias in the developed metrics for our label propagation methods, the label propagation results were also evaluated using two state of the art methods: Probabilistic Rand Index [129] and *tqm* [63]. Results from these metrics were compared with the results of our evaluation metrics. It was observed that evaluation scores from spatial consistency metric and temporal consistency metric followed a trend similar to the evaluation scores of the Probabilistic Rand Index and *tqm* metrics respectively.

8.2 Future work

Spatio-temporal displacement constraint: Spatio-temporal displacement constraint allows us to achieve labelling with lesser number of pixel to label comparisons during each iteration of the α-expansion in comparison to unconstrained α-expansion based methods. Generally in a video sequence, objects moves with different speeds. Therefore, their frame to frame displacements would be different. It should be possible to further restrict the comparisons based on a prior expectations of displacements, based on the displacement of the regions in previous frames. This would further reduce the number of pixel to label comparisons and consequently would produce more robust results.

Handling appearance of new objects: The developed algorithm is only able to accommodate for the appearance of new objects if the colour distributions of the new objects are different from colour distributions of their neighbours. The algorithm will mis-classify pixels belonging to the new objects if colour distributions of the new objects and their neighbouring objects are similar. One possible improvement would be to include more properties of objects in the algorithm, such as texture.

Improving label propagation results using both backward and forward propagation: Our label propagation algorithm processes the video in the forward direction where labels existing in previous frames are propagated to label each pixel of the current frame. It was shown that over time, the label propagation method learns more about the *segments*, consequently improving the segmentation results. It would be interesting to explore how segmentation results of the current frame can be used to improve the segmentation results of previous frames.

Learning the physical explanations behind changes and occlusion boundaries: The developed algorithms learned the physical explanations – occlusion and deocclusion – behind any change in the shape of *segments*. In future, these explanations can be explored to understand the 3D structure of the objects represented by the *segments* and ultimately the scene structure.

Evaluation metrics: Our temporal consistency metric computes the temporal consistency score for boundary pixels of the current frame by analysing all possible sets of corresponding pixels, i.e. *paths*, from a block of frames. It was shown in chapter 6 that a maximum of M^{n-1} *paths* can be created for any boundary pixel of the current frame where M is the maximum number of boundary pixels that can be present in the spatio-temporal neighbourhood and n is the number of frames in the block. Due to a large number of paths under consideration for each pixel of the current frame, time complexity of our temporal consistency metric is very large and of the order of $O(M^{n-1})$. It would be interesting to explore how the number of *paths* can be reduced to minimise the overall time complexity of our algorithm. One possibility is to use the *path* information of the neighbouring boundary pixels.

Bibliography

[1] T. Wang and J. Collomosse, "Probabilistic motion diffusion of labeling priors for coherent video segmentation," *Multimedia, IEEE Transactions on*, vol. 14, no. 2, pp. 389–400, 2012.

[2] A. N. Stein and M. Hebert, "Occlusion boundaries from motion: Low-level detection and mid-level reasoning," *International Journal of Computer Vision*, vol. 82, no. 3, pp. 325–357, 2009.

[3] S. Khan and M. Shah, "Object based segmentation of video using color, motion and spatial information," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2, pp. II–746–II–751, 2001.

[4] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp. 416–423, Jul. 2001.

[5] E. Galmar and B. Huet, "Graph-based spatio-temporal region extraction," in *ICIAR 2006, 3rd International Conference on Image Analysis and Recognition, Portugal; Also published as Lecture Notes in Computer Science (LNCS) Volume 4141, 2006*, Sep. 2006.

[6] Y. Wang, K. Loe, T. Tan, and J. K. Wu, "Spatiotemporal video segmentation based on graphical models," *Image Processing, IEEE Transactions on*, vol. 14, pp. 937–947, Jul. 2005.

[7] L. Zhang and Q. Ji, "Segmentation of video sequences using spatial-temporal conditional random fields," in *Motion and video Computing, 2008. WMVC 2008. IEEE Workshop on*, pp. 1–7, Jan. 2008.

[8] M. El Hassani, S. Jehan-Besson, L. Brun, M. Revenu, M. Duranton, D. Tschumperlé, and D. Rivasseau, "A time-consistent video segmentation algorithm designed for real-time implementation," *VLSI Design*, vol. 2008, no. 2, pp. 1–12, 2008.

[9] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph-based video segmentation," in *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*, pp. 2141–2148, Jun. 2010.

[10] L. Silva and J. Scharcanski, "Video segmentation based on motion coherence of particles in a video sequence," *Image Processing, IEEE Transactions on*, vol. 19, pp. 1036–1049, Apr. 2010.

[11] M. Kumar, P. Torr, and A. Zisserman, "Learning layered motion segmentations of video," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 33–40, Oct. 2005.

[12] V. Badrinarayanan, F. Galasso, and R. Cipolla, "Label propagation in video sequences," in *Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on*, pp. 3265–3272, Jun. 2010.

[13] I. Budvytis, V. Badrinarayanan, and R. Cipolla, "Label propagation in complex video sequences using semi-supervised learning," in *Proceedings of the British Machine Vision Conference*, pp. 27.1–27.12, BMVA Press, 2010.

[14] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov, "Bilayer segmentation of live video," in *Computer Vision and Pattern Recognition, 2006. CVPR 2006. IEEE Computer Society Conference on*, vol. 1, pp. 53–60, Jun. 2006.

[15] A. Chen and J. Corso, "Propagating multi-class pixel labels throughout video frames," in *Image Processing Workshop (WNYIPW), 2010 Western New York*, pp. 14–17, Nov. 2010.

[16] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 1222–1239, Nov. 2001.

[17] P. Kohli, M. Kumar, and P. Torr, "P3 beyond: Solving energies with higher order

cliques," in *Computer Vision and Pattern Recognition, 2007. CVPR 2007. IEEE Conference on*, pp. 1–8, Jun. 2007.

[18] P. Sturgess, K. Alahari, L. Ladicky, and P. H. S. Torr, "Combining appearance and structure from motion features for road scene understanding," in *Proceedings of the British Machine Vision Conference*, pp. 62.1–62.11, BMVA Press, 2009.

[19] H. Winnemoller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, (New York, NY, USA), pp. 1221–1226, ACM, 2006.

[20] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," in *Sixth International Conference on Computer Vision*, pp. 1154–1160, 1998.

[21] C. Marc, P. Stephane, and N. Henri, "Segmentation of non-rigid video objects using long term temporal consistency," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 2, pp. II–93 – II–96, 2002.

[22] Y. Huang, Q. Liu, and D. Metaxas, "Video object segmentation by hypergraph cut," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 1738–1745, Jun. 20–25 2009.

[23] L. Ladicky, C. Russell, P. Kohli, and P. Torr, "Inference methods for crfs with co-occurrence statistics," *International Journal of Computer Vision, ECCV special award issue*, 2011.

[24] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[25] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, pp. 1101–1113, Nov. 1993.

[26] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *Computer Vision and Pattern Recognition, 1997. Proceedings. 1997 IEEE Computer Society Conference on*, pp. 731–737, Jun. 17–19 1997.

[27] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, vol. 1, pp. 10–17, Oct. 13–16 2003.

[28] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "Turbopixels fast superpixels using geometric flows," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 2290–2297, Dec. 2009.

[29] S. Yu, "Segmentation using multiscale cues," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–247–254, Jun. 27 – Jul. 2 2004.

[30] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *Computer Vision and Pattern Recognition. CVPR 2005. Proceedings of the 2005 IEEE Computer Society Conference on*, vol. 2, (Washington, DC, USA), pp. 1124–1131, IEEE Computer Society, 2005.

[31] A. Ducournau, S. Rital, A. Bretto, and B. Laget, "A multilevel spectral hypergraph partitioning approach for color image segmentation," in *Signal and Image Processing Applications (ICSIPA), 2009 IEEE International Conference on*, pp. 419–424, Nov. 18–19 2009.

[32] R. Nock and F. Nielsen, "Statistical region merging," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1452–1458, Nov. 2004.

[33] G. Nee, S. Jehan-Besson, L. Brun, and M. Revenu, "Significance tests and statistical inequalities for region matching," *Structural, Syntactic, and Statistical Pattern Recognition*, pp. 350–360, 2008.

[34] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 530–549, May 2004.

[35] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie, "Beyond pairwise clustering," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 838–845, Jun. 2005.

[36] D. Zhou, J. Huang, and B. Scholkopf, "Learning with hypergraphs: Clustering, classification, and embedding," *Advances in Neural Information Processing Systems*, vol. 19, p. 1601, 2007.

[37] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in vlsi domain," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 7, pp. 69–79, Mar. 1999.

[38] M. Irani and P. Anandan, "About direct methods," in *Vision Algorithms: Theory and Practice* (B. Triggs, A. Zisserman, and R. Szeliski, eds.), vol. 1883 of *Lecture Notes in Computer Science*, pp. 267–277, Springer Berlin / Heidelberg, 2000.

[39] P. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Vision Algorithms: Theory and Practice, number 1883 in LNCS*, pp. 278–295, Springer-Verlag, 2000.

[40] D. Zhong and S.-F. Chang, "Long-term moving object segmentation and tracking using spatio-temporal consistency," in *Image Processing, 2001. Proceedings. 2001 International Conference on*, vol. 2, pp. 57–60, Oct. 7–10 2001.

[41] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, Dec. 2006.

[42] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 48, no. 3, pp. 259–302, 1986.

[43] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-6, pp. 721–741, Nov 1984.

[44] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1, pp. 105–112, 2001.

[45] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 1124 –1137, Sep. 2004.

[46] A. Delong, A. Osokin, H. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2173–2180, Jun. 2010.

[47] K. Alahari, P. Kohli, and P. Torr, "Dynamic hybrid algorithms for map inference in discrete mrfs," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 1846–1857, Oct. 2010.

[48] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, pp. 147–159, Feb. 2004.

[49] T. Wang, J. Guillemaut, and J. Collomosse, "Multi-label propagation for coherent video segmentation and artistic stylization," in *Image Processing (ICIP), 2010. 17th IEEE International Conference on*, pp. 3005–3008, Sep. 2010.

[50] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[51] Y. Weiss and W. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *Information Theory, IEEE Transactions on*, vol. 47, pp. 736–744, Feb. 2001.

[52] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, pp. 1068–1080, Jun. 2008.

[53] P. Felzenszwalb and D. Huttenlocher, "Efficient belief propagation for early vision," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–261–I–268, Jun. 27–Jul. 2 2004.

[54] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[55] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society Series B Methodological*, vol. 51, no. 2, pp. 271–279, 1989.

[56] B. Kwolek, "Object segmentation in video via graph cut built on superpixels," *Fundamenta Informaticae*, vol. 90, pp. 379–393, Dec. 2009.

[57] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient nd image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 2006.

[58] Kohli, Pushmeet and Ladický, L'ubor and Torr, Philip, "Robust higher order potentials for enforcing label consistency," *International Journal of Computer Vision*, vol. 82, pp. 302–324, 2009.

[59] Y. Zhang, "A survey on evaluation methods for image segmentation," *Pattern Recognition*, vol. 29, no. 8, pp. 1335–1346, 1996.

[60] C. 211quat, "Redundancy reduction techniques and content analysis for multimedia services," in *COST project.*

[61] C. 211quat, "Call for am comparisonsŮcompare your segmentation algorithm to the cost 211quat analysis model," in *COST project.*

[62] E. Gelasca and T. Ebrahimi, "On evaluating video object segmentation quality: A perceptually driven objective metric," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 3, no. 2, pp. 319–335, 2009.

[63] M. Wollborn and R. Mech, "Procedure for objective evaluation of vop generation algorithms," *Doc. ISO/IEC JTC1/SC29/WG11 MPEG97/2704, Fribourg*, 1997.

[64] P. Correia and F. Pereira, "Objective evaluation of relative segmentation quality," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 1, pp. 308–311, 2000.

[65] P. Correia and F. Pereira, "Objective evaluation of video segmentation quality," *Image Processing, IEEE Transactions on*, vol. 12, no. 2, pp. 186–200, 2003.

[66] X. Marichal and P. Villegas, "Objective evaluation of segmentation masks in video sequences," in *Proc. X European Signal Processing Conference*, vol. 4, Sep. 2000.

[67] P. Villegas and X. Marichal, "Perceptually-weighted evaluation criteria for segmentation masks in video sequences," *Image Processing, IEEE Transactions on*, vol. 13, no. 8, pp. 1092–1103, 2004.

[68] J. Nascimento and J. S. Marques, "New performance evaluation metrics for object detection algorithms," in *Proceedings of IEEE PETS Workshop*, 2004.

[69] F. Monteiro and A. Campilho, "Performance evaluation of image segmentation," *Image Analysis and Recognition*, pp. 248–259, 2006.

[70] E. Gelasca, T. Ebrahimi, M. Karaman, and T. Sikora, "A framework for evaluating video object segmentation algorithms," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pp. 198–198, 2006.

[71] S. E. Maxwell and H. D. Delaney, *Designing Experiments and Analyzing Data: A Model Comparison Perspective, Second Edition.* Routledge Academic, 2 ed., May 2003.

[72] C. Erdem and B. Sankur, "Performance evaluation metrics for object-based video segmentation," in *Proc. X European Signal Processing Conference*, vol. 2, pp. 917–920, Sep. 5–8 2000.

[73] C. Erdem, A. Murat Tekalp, and B. Sankur, "Metrics for performance evaluation of video object segmentation and tracking without ground-truth," in *Image Processing, 2001. Proceedings. International Conference on*, vol. 2, pp. 69–72, 2001.

[74] P. L. Correia and F. Pereira, "Stand-alone objective segmentation quality evaluation," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, no. 4, p. 431748, 2002.

[75] C. Erdem, B. Sankur, and A. Tekalp, "Performance measures for video object segmentation and tracking," *Image Processing, IEEE Transactions on*, vol. 13, no. 7, pp. 937–951, 2004.

[76] R. Mech and F. Marqués, "Objective evaluation criteria for 2d-shape estimation results of moving objects," *EURASIP Journal on Applied Signal Processing*, vol. 2002, pp. 401–409, Apr. 2002.

[77] A. Senior, A. Hampapur, Y. li Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," in *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, 2001.

[78] R. B. Potts and C. Domb, "Some generalized order-disorder transformations," *Proceedings of the Cambridge Philosophical Society*, vol. 48, p. 106, 1952.

[79] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr, "Interactive image segmentation using an adaptive gmmrf model," *Computer Vision - ECCV 2004*, pp. 428–441, 2004.

[80] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, pp. 1151–1163, July 2002.

[81] P. Ganesan, V. Rajini, and R. Rajkumar, "Segmentation and edge detection of color images using cielab color space and edge detectors," in *Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on*, pp. 393–397, Dec 2010.

[82] M. Tuceryan and A. K. Jain, "Texture analysis," *Handbook of pattern recognition and computer vision*, vol. 2, pp. 207–248, 1993.

[83] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-3, pp. 610–621, Nov 1973.

[84] M. Tuceryan and A. Jain, "Texture segmentation using voronoi polygons," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, pp. 211–216, Feb 1990.

[85] H. Voorhees and T. Poggio, "Detecting textons and texture boundaries in natural images," in *Proceedings of the First International Conference on Computer Vision*, pp. 250–258, 1987.

[86] T. Lindeberg, "Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention," *International Journal of Computer Vision*, vol. 11, pp. 283–318, dec 1993.

[87] G. R. Cross and A. K. Jain, "Markov random field texture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-5, pp. 25–39, Jan 1983.

[88] H. Derin and H. Elliott, "Modeling and segmentation of noisy and textured images using gibbs random fields," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, pp. 39–55, Jan 1987.

[89] B. B. Mandelbrot, *The fractal geometry of nature*. San Francisco: Freeman, 1982.

[90] J. M. Keller, S. Chen, and R. M. Crownover, "Texture description and segmentation through fractal geometry," *Computer Vision, Graphics, and Image Processing*, vol. 45, no. 2, pp. 150–166, 1989.

[91] B. J. Super and A. C. Bovik, "Localized measurement of image fractal dimension using gabor filters," *Journal of Visual Communication and Image Representation*, vol. 2, no. 2, pp. 114–128, 1991.

[92] F. Schroff, A. Criminisi, and A. Zisserman, "Single-histogram class models for image segmentation," in *Indian Conference on Computer Vision, Graphics and Image Processing*, 2006.

[93] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," *Context*, vol. 3951, no. 1, pp. 1–14, 2006.

[94] J. Winn, A. Criminisi, and T. Minka, "Object categorization by learned universal visual dictionary," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2, pp. 1800–1807, Oct. 2005.

[95] T. Leung and J. Malik, "Representing and recognizing the visual appearance of materials using three-dimensional textons," *International Journal of Computer Vision*, vol. 43, pp. 29–44, Jun. 2001.

[96] M. A. Hoang, J.-M. Geusebroek, and A. W. Smeulders, "Color texture measurement and segmentation," *Signal Processing*, vol. 85, no. 2, pp. 265–275, 2005.

[97] G. M. Johnson and M. D. Fairchild, "A top down description of s-cielab and ciede2000," *Color Research & Application*, vol. 28, no. 6, pp. 425–435, 2003.

[98] X. He, R. Zemel, and D. Ray, "Learning and incorporating top-down cues in image segmentation," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science*, pp. 338–351, Springer Berlin Heidelberg, 2006.

[99] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, pp. 898–916, May 2011.

[100] A. Rabinovich, S. Belongie, T. Lange, and J. Buhmann, "Model order selection and cue combination for image segmentation," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 1130–1137, 2006.

[101] D. Hoiem, A. Efros, and M. Hebert, "Geometric context from a single image," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 654–661, 2005.

[102] B. Russell, W. Freeman, A. Efros, J. Sivic, and A. Zisserman, "Using multiple segmentations to discover objects and their extent in image collections," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 1605–1614, 2006.

[103] T. H. Kim, K.-M. Lee, and S.-U. Lee, "Nonparametric higher-order learning for interactive segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3201–3208, 2010.

[104] A. N. Stein and M. Hebert, "Using spatio-temporal patches for simultaneous estimation of edge strength, orientation, and motion," *Computer Vision and Pattern Recognition Workshop*, vol. 0, p. 19, 2006.

[105] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[106] *PETS 2001 Benchmark Data.* Online, 2001.

[107] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6315 of *Lecture Notes in Computer Science*, pp. 282–295, Springer Berlin Heidelberg, 2010.

[108] M. Styner, C. Brechbuhler, G. Szckely, and G. Gerig, "Parametric estimate of intensity inhomogeneities applied to mri," *Medical Imaging, IEEE Transactions on*, vol. 19, no. 3, pp. 153–165, 2000.

[109] T. Papadimitriou, K. Diamantaras, M. Strintzis, and M. Roumeliotis, "Video scene

segmentation using spatial contours and 3-D robust motion estimation," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 14, no. 4, pp. 485–497, 2004.

[110] B. Jian and B. Vemuri, "Robust point set registration using gaussian mixture models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, pp. 1633–1645, Aug. 2011.

[111] D. Rueckert and P. Aljabar, "Nonrigid registration of medical images: Theory, methods, and applications [applications corner]," *Signal Processing Magazine, IEEE*, vol. 27, no. 4, pp. 113–119, 2010.

[112] N. Paragios and R. Deriche, "Geodesic active contours and level sets for the detection and tracking of moving objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 3, pp. 266–280, 2000.

[113] C. Tomasi and T. Kanade, *Detection and tracking of point features*. School of Computer Science, Carnegie Mellon Univ., 1991.

[114] P. Sand and S. Teller, "Particle video: Long-range motion estimation using point trajectories," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2195 –2202, 2006.

[115] B. Wu and R. Nevatia, "Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors," *International Journal of Computer Vision*, vol. 75, no. 2, pp. 247–266, 2007.

[116] H. Park, G. Martin, and A. Bhalerao, "Local affine image matching and synthesis based on structural patterns," *Image Processing, IEEE Transactions on*, vol. 19, no. 8, pp. 1968–1977, 2010.

[117] L. Lucchese, "A frequency domain technique based on energy radial projections for robust estimation of global 2d affine transformations," *Computer Vision and Image Understanding*, vol. 81, no. 1, pp. 72–116, 2001.

[118] V. Traver and F. Pla, "Motion analysis with the radon transform on log-polar images," *Journal of Mathematical Imaging and Vision*, vol. 30, no. 2, pp. 147–165, 2008.

[119] A. Kadyrov and M. Petrou, "Affine parameter estimation from the trace transform," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1631–1645, 2006.

[120] J. Lewis, "Fast normalized cross-correlation," in *Vision interface*, vol. 10, pp. 120–123, 1995.

[121] R. Kwitt and A. Uhl, "Image similarity measurement by kullback-leibler divergences between complex wavelet subband statistics for texture retrieval," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 933–936, 2008.

[122] J. Goldberger, S. Gordon, and H. Greenspan, "An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, vol. 1, pp. 487–493, 2003.

[123] J. Hershey and P. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–317–IV–320, 2007.

[124] F. Moscheni and F. Dufaux, "Regions merging based on robust statistical testing," in *Visual Communications and Image Processing*, pp. 1118–1129, International Society for Optics and Photonics, 1996.

[125] J. Wang and E. Adelson, "Representing moving images with layers," *Image Processing, IEEE Transactions on*, vol. 3, pp. 625–638, Sep 1994.

[126] L. Torres, D. Garcia, and A. Mates, "A robust motion estimation and segmentation approach to represent moving images with layers," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 4, pp. 2981–2984 vol.4, Apr 1997.

[127] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes, "Layered object detection for multi-class segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3113–3120, IEEE, 2010.

[128] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes, "Layered object models for

image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 1731–1743, Sep 2012.

[129] R. Unnikrishnan and M. Hebert, "Measures of similarity," in *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, vol. 1, pp. 394–394, IEEE, 2005.