Handling Transparency in Digital Video

A dissertation submitted to the University of Dublin
for the degree of Doctor of Philosophy

Mohamed A. Elgharib
Trinity College Dublin, December 2011

Signal Processing and Media Applications
Department of Electronic and Electrical Engineering
Trinity College Dublin

To Ethics

Abstract

This thesis is concerned with handling transparency in digital video. We define transparency as a mixture between two different distinctive layers. This thesis has three main contributions.

The first contribution is an approach for removing two common types of degradations on Digital Video known as 'Blotches' and 'Line Scratches'. Blotches are impulsive dark and bright spots distributed randomly over an image sequence. Line scratches are temporally consistent dark and bright vertical lines propagating through an image sequence. Current removal techniques model such corruptions as an opaque foreground layer superimposed on the original (background) 'clean' layer. This often generates restoration artifacts due to underestimation and/or overestimation of the corruption region. We propose a removal technique which models corruptions as a semi-transparent layer superimposed on the original 'clean' layer. Removal is then achieved by estimating corruption opacity and the underlaying original data. We generate a solution using a Bayesian framework and we use novel spatial and temporal priors. Restoration results are compared against ground truth estimates. Ground truth estimates are derived from IR scans of corruptions. Restoration results show that our restoration technique generates more accurate estimation of the corruption borders over previous work. This generates better removal despite texture and motion complexity.

The second contribution of this thesis is a technique for automated detection of reflections in image sequences. Regions of reflections are common in video and they are often the result of superimposing a semi-transparent foreground layer over a background layer. This phenomenon causes many image processing techniques to fail as they assume the presence of one layer at each examined site e.g. motion estimation and object recognition. This calls for the need of an automated technique which detects such regions and assigns different treatments to them. However, as reflections can result by mixing any two images, they can come in different forms and colors. This makes their detection a hard problem that was not addressed before. We propose a technique for automated detection of reflections by analyzing feature point trajectories. We examine several spatial and temporal features. This generates a set of weak detectors. A strong detector is generated by combining the weak detectors. We generate a solution using a Machine Learning framework, impose spatial and temporal smoothness on the generated masks and our results show high reflection detection rate with rejection to regions of complicated motion.

The third contribution of this thesis is a technique for multiple motion estimation in regions of reflections. As such regions have two layers superimposed over each other, they usually have two motions per pel, one for each layer. Most motion estimators assume the presence of one motion per examined site, an assumption that is violated in reflections. Current multiple motion estimators assume constant motion over at least three frames. As a result they can not handle non-uniform motions as ones arising due to camera shake or motion acceleration. We present an approach for multiple motion estimation based on the observation that the motion for a specific

foreground/background layer can be used to temporally align this layer and then separate it from the examined mixture. Based on this observation we model the correct motions of an examined reflection as the ones generating the best reconstructions of the foreground and background layers at the examined site. A solution is generated within a Bayesian framework and we use novel temporal priors. Results show better multiple motion estimation over previous work with handling of strong motion inconsistencies. However, our technique is only designed to handle regions of reflections and hence generates erroneous measurements in regions not containing reflections. We therefore show how to use reflection detection masks to weight out erroneous measurements in regions not containing reflection. These masks are generated by our reflection detection technique and results show rejection of regions not containing reflections from the final motion estimates.

Declaration

I hereby declare that this thesis has not been submitted as an exercise for a degree at this or any other University and that it is entirely my own work.

I agree that the Library may lend or copy this thesis upon request.

Signed,

Mohamed A. Elgharib

December 20, 2011.

Acknowledgments

First of all I would like to thank Prof. Anil Kokaram for introducing me to the world of Image Processing and Computer Vision. Before meeting him I did not know that such world even existed. I owe him a lot! Joint thanks goes to my supervisors Prof. Anil Kokaram and Dr. François Pitié for their (to name a few) continuous support, advice, inspiration and encouragement throughout all the years I worked in image processing. I also would like to thank my old Egyptian university, Cairo University, for introducing me to Digital Signal Processing.

Thanks to all past and present members of Sigmedia research group. I particularly wish to thank Dr. Gary Baugh for sharing ideas (and for teaching me Jamaican, *yaaa maaan*), Ken Sooknanan, Dr. David Corrigan, Kangyu Pan, Andrew Rankin, Craig Berry, Félix Raimbault, Finnian Kelly, Luca Cappelletta, Róisín Rowley-Brooke, Andrew Hines, Dr. Rozenn Dahyot, Dr. Claire Masterson, Dr. Dan Ring, Dr. Deirdre O'Regan, Dr. Damien Kelly and Yun Feng. I also would like to thank all the staff of Electronic and Electrical Engineering (EEE) Department at Trinity College Dublin. Special mention to Dr. Martin Burke, Bernadette Clerkin, Robbie Dempsey, Conor Nolan, Teresa Lawlor and Shane.

My PhD study would have not been possible without the generous funding that I received from the Irish Research Council for Science, Engineering and Technology (IRCSET). Thanks a lot! I also would like to thank Science Foundation Ireland (SFI) and Adobe Systems for funding my publications. Furthermore, I would like to thank all the Irish people for hosting me in their lovely country.

My family have always been supportive of me especially in pursuing a PhD degree in Engineering. In particular I would like to thank my Mom, Dad, Sister and Brother.

I also would like to thank my friends back in Egypt who have always supported me in pursuing a PhD. I am sure they all will become big in the near future. In particular I would like to mention Ahmed Selim, Mohamed Abouzied, Hatem Osman, Mohamed Alaa, Karim Yehia, Amr Shahat, Amr Abdulzahir, Alhassan Khedr, Ahmed Abdulkareem, Ahmed Helmy and Ahmed Bayoumy.

Contents

List of Acronyms

SSD Sum of Square Dfifference

DFD Displaced Frame Difference

DPD Displaced Pixel Difference

ICM Iterated Conditional Modes

QPBO Quadratic Pseudo-Boolean Optimization

MAP Maximum *a posteriori*

ML Maximum Likelihood

MSE Mean Squared Error

ROC Receiver Operating Characteristic

SSIM Structural Similarity

IR Infrared

MRF Markov Random Field

Pel Pixel or Picture element

SDI Spike Detection Index

BTCR Bayesian Transparent Corruption Remover

BTBR Bayesian Transparent Blotch Remover

BTBR-S Bayesian Transparent Blotch Remover - Spatial

BTBR-T Bayesian Transparent Blotch Remover - Temporal

BTBR-F Bayesian Transparent Blotch Remover - Fusion

BTLR Bayesian Transparent Line Remover

FEAPARD Feature Point Analyses for Reflection Detection

FEAPARD-S Feature Point Analyses for Reflection Detection - Spatial

FEAPARD-T Feature Point Analyses for Reflection Detection - Temporal

FEAPARD-F Feature Point Analyses for Reflection Detection - Final

BIMS Bayesian Inference for Multiple Motion Estimation through Layer Separation

JONDI Joint Noise Reduction, Detection and Interpolation

AR Autoregressive

OFCE Optical Flow Constraint Equation

GNGC Generalized Normalized Gray-scale Correlation

1 Introduction

Digital video and film manipulation has been the cornerstone of digital video broadcast and digital cinema industry for some time now. One of the major new advances in recent years has been a better understanding of images in sequences. This has allowed video processing tools to be increasingly used for postproduction and consumer video applications e.g. restoration, object cut out and frame rate conversion. However transparency in video still remains a challenge for all video processing tools and prevents automated tools from being used ubiquitously for video editing and inference. For example, in the vast literature on frame rate conversion [15, 92], motion compensated interpolation is a key component. When faced with transparency however, new frames cannot be built reliably using the usual assumption of only one moving object being observed. Figure 1.1 shows such example of a reflection of a building against paper posters. Archive restoration provides another good example. In that industry, automated *dust busting* has become important for removing dirt and sparkle (missing data or blotches) from degraded image sequences. Figures 1.2 and 1.3 show examples from degraded sequences exhibiting both dirt and line scratches respectively. In this problem, the corruption is in fact transparent. Traditional dust busting strategies tended to ignore this and that meant that errors in detecting the degradation were very visible since *hard* rather than *soft* decisions were being taken at each pixel site. Even motion estimation itself is affected by transparency, since the usual assumption of one motion per pixel breaks down in the presence of reflections or shadows.

Traditionally problems due to transparency have been handled by allowing for some kind of fallback or error mode in many video processing systems e.g. pathological motion detection [11, 19, 63]. In this thesis however, we address the phenomenon of transparency explicitly. We

Figure 1.1: *In clockwise direction: Frames 1, 10, 20 from a sequence containing reflection of a building superimposed on paper posters. In this sequence the building is moving to the left while the posters are moving to the right. This generates two motions per pel.*

first consider blotch and line removal using transparency as a corruption model and then we move on to address the larger problems of transparency/reflection detection and multiple motion handling. The connecting thread in this work is the model we employ to express transparency. Given the examples in figures 1.1 and 1.2-1.3, we express image patches exhibiting transparency as a mixture of two layers. For a site $\mathbf{x}$, the resulting mixture M can be written as a linear combination between the foreground F and background B layers as follows

$$M(\mathbf{x}) = \alpha(\mathbf{x})F(\mathbf{x}) + (1 - \alpha(\mathbf{x}))B(\mathbf{x}) \tag{1.1}$$

Here $\alpha(\mathbf{x})$ denote the foreground visibility at site $\mathbf{x}$. Complete obliteration of the background

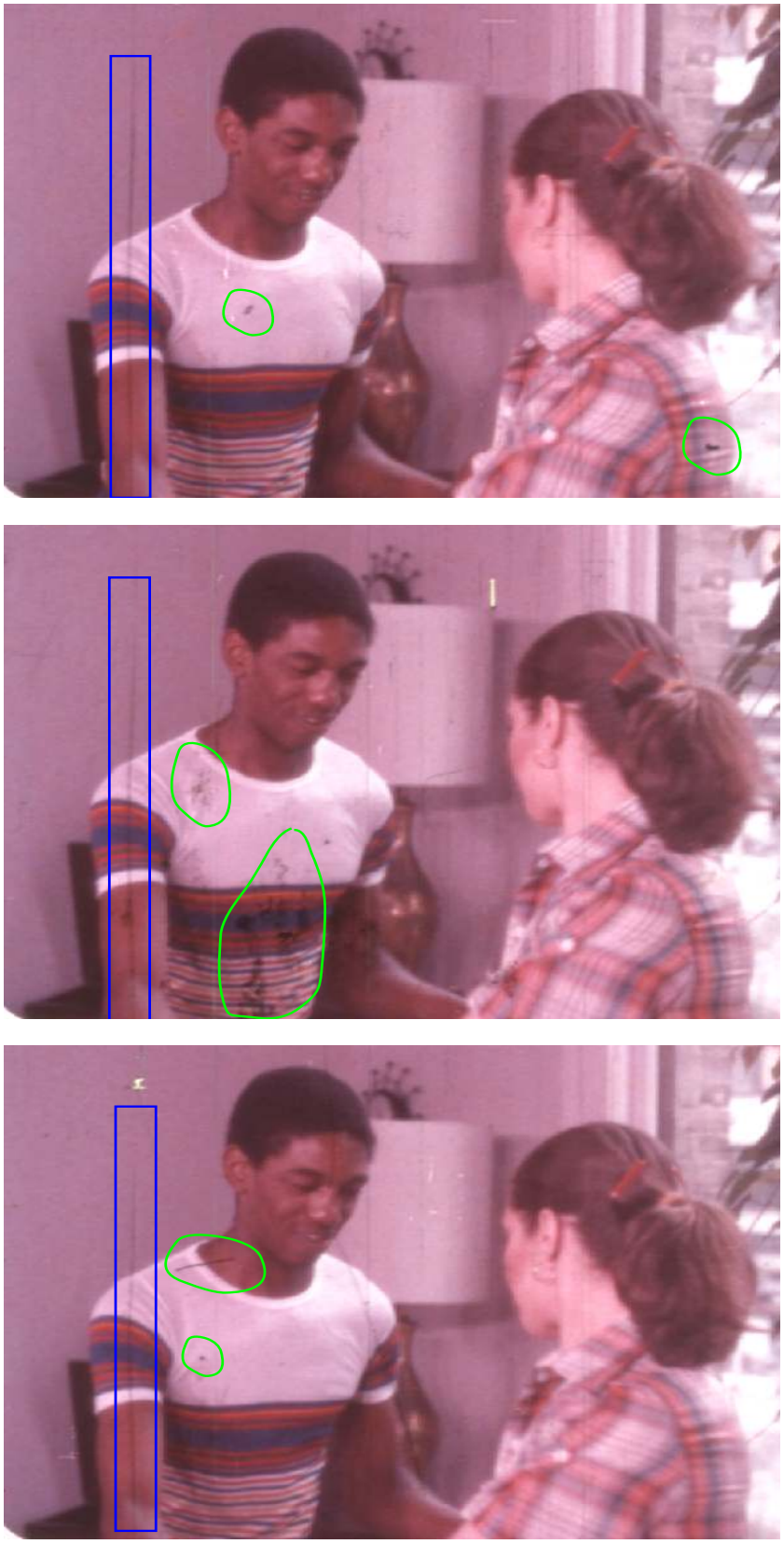


Figure 1.2: *From top: Three consecutive corrupted frames. Blotches and line scratches are shown in green and blue respectively. Botches are spatially and temporally impulsive events while line scratches are temporally consistent events. Images Courtesy of Institut national de l'audiovisuel.*

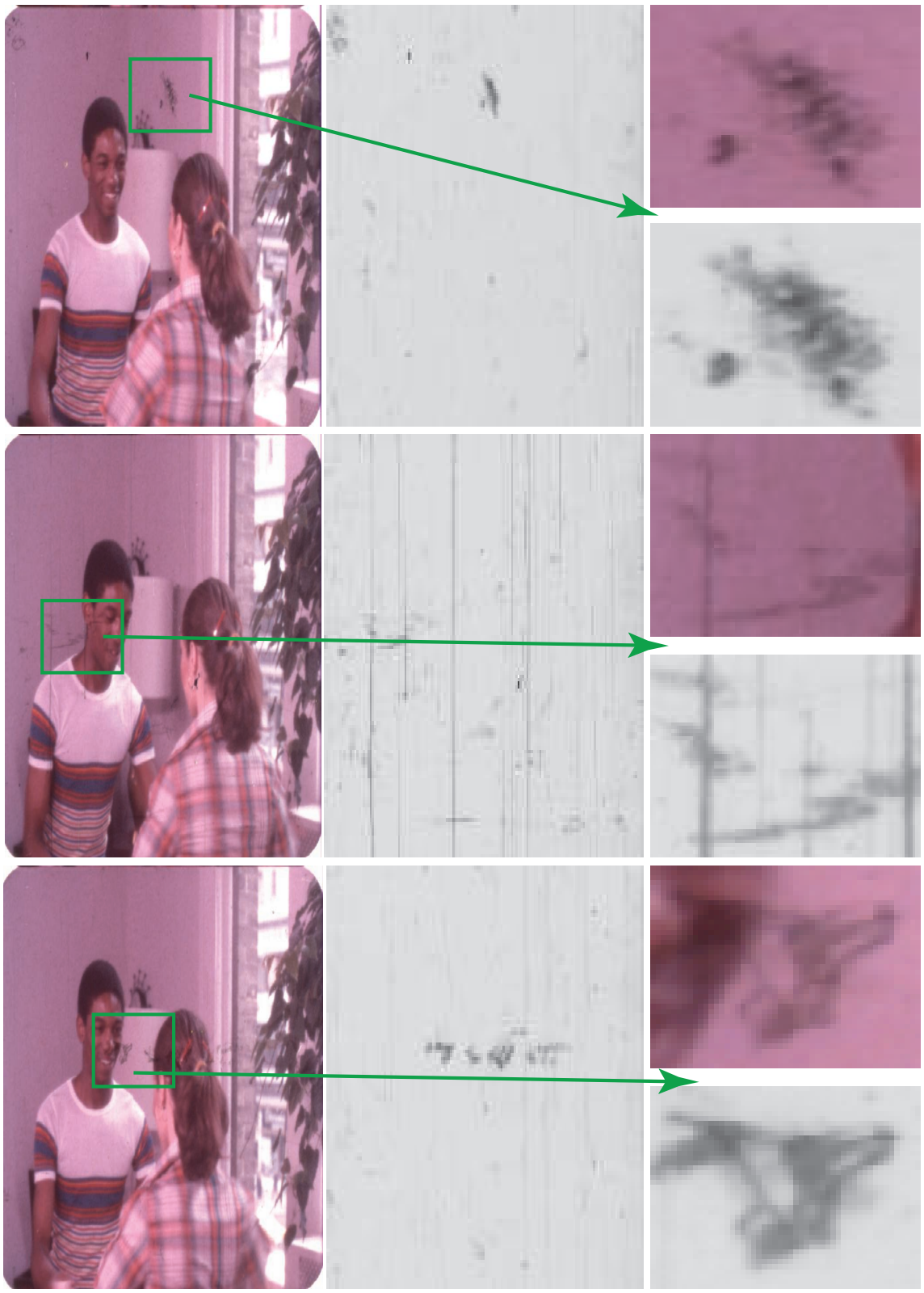


Figure 1.3: *Each row shows a different frame from a corrupted image sequence and its corresponding IR scan of corruption in the middle column. The IR scan is bright in clean regions and dark in corrupted regions. The last column shows the green areas zoomed on. Different IR gray values for corruptions show that corruptions are semi-transparent dirt layers.*

layer is therefore represented by $\alpha(\mathbf{x}) = 1$. For blotch and line scratches removal $\alpha(\mathbf{x})$ represents corruption opacity. $\alpha(\mathbf{x})$ estimation is an important step in layer separation. Layer separation analysis of natural images is performed to infer the presence of multiple layers for the purpose of reflection detection.

This thesis proposes automated approaches to handle transparency in digital video. In the first main contribution of this thesis we use equation. 1.1 to model degradations as a foreground corruption layer linearly mixed with the original 'clean' background layer. Removal is then performed by reconstructing the underlying original data. This is done by estimating the corruption opacity values α together with the original background data B. The problem of estimating the opacity values for mixtures of natural images is well studied [8, 9, 17, 30, 31, 33, 36, 52, 58, 62, 64–66, 70, 76, 82, 90, 91, 96]. Here the name 'Matte' is used in the literature to describe the estimated opacity values. Chuang et al. [17] proposed a Bayesian solution for the matte extraction problem. Their technique solves for the background data in an implicit form during the estimation of α. We use this approach to estimate the background original clean data B as well as the corruption opacity α. We generate a solution within a Bayesian framework and we use novel spatial and temporal priors. Restoration results show better estimation of the corruption borders and corruption removal robust to texture and motion complexity.

The second contribution of this thesis is concerned with the detection of transparencies/reflections in natural image sequences (see figure. 1.1). Reflections are modeled as a mixture between a foreground layer superimposed on a background layer. Detection is achieved by analyzing feature point trajectories. Several weak detectors are proposed, some of which flag regions of reflection as ones generating high motion discontinuities. Others flag reflections as regions where the examined site can be separated into two different foreground and background layers. Separation here is done by employing some ideas used in current reflection separation techniques [12, 16, 22, 26, 39–42, 45, 53, 54, 71, 73, 74, 77, 94, 95, 97, 98]. Note that here we are not interested in state of the art layer separation, instead we are just interested in separation that is good enough for reflection detection. The final reflection detector is generated as a combination of the weak detectors and results show high detection rate with rejection to complicated motion.

The final contribution of this thesis is to propose a technique to estimate the underlying foreground and background motions in regions of reflections. This technique is based on Weiss work on reflection separation [94] where he showed that one can extract the foreground or the background layer of a reflection once this layer is temporally aligned through an image sequence. Hence we model the correct motion of the background layer as the one generating the best temporal alignment of this layer and hence generating the best background estimate. The foreground motion is estimated using the same approach. Here feature point trajectories act as pool of candidates for the motions. Unlike previous work on multiple motion estimation, our technique is robust to temporal motion inconsistencies. As a final note we show how reflection detection masks can be used to weight out erroneous motion estimates generated in regions

not containing reflections. Those masks are generated from our proposed reflection detection technique.

1.1 Thesis outline

A brief summary of each chapter is as follows.

Chapter 2 Handling Transparency in Natural Images: Related Work

This chapter discusses previous work that is related to the three main contributions of this thesis. We discuss three main problems. The first is the extraction of opacity mattes in natural images. Note however this thesis does not address the problem of matte extraction in an explicit form. Instead, it just uses ideas from the matting problem. Hence in this chapter we focus on discussing the matting technique of Chuang et al. [17] as it is used for the corruption removal technique proposed in this thesis. The second problem addressed in this chapter is the separation/reconstruction of the foreground and background layers from reflections. Again, as this thesis does not address the problem of layer separation in an explicit form, we only focus on separation ideas that are used in the main contributions of this thesis. The last problem addressed in this chapter is the estimation of the foreground and background motions for reflections. Here we present a literature review of current multiple motion estimators.

Chapter 3: Blotch and Line Scratch Removal: A Review

This chapter describes previous work on removing blotches and line scratches. Current techniques largely fall into two main categories. The first models corruptions as an opaque foreground layer superimposed on the original background. Here the mixing opacity α in equation 1.1 takes only binary values where $\alpha = 1$ represents complete obliteration of the background layer while $\alpha = 0$ represents uncorrupted region. The second category models corruptions as semi-transparent layers superimposed on the original layer. Here the mixing opacity α in equation 1.1 takes non binary values.

Chapter 4: Bayesian Inference for Semi-transparent Blotch and Line Removal

This chapter proposes a new approach for removing blotches and line scratches from image sequences. Here we model the corrupted frame as a mixture between the corruption layer and the original layer. Chuang et al. [17] matte extraction technique is used to estimate the corruption opacity as well as the original data. A solution is presented within a Bayesian framework and we use novel spatial and temporal priors. The proposed blotch removal technique is called BTBR short for **B**ayesian **T**ransparent **B**lotch **R**emover while the proposed line removal technique is called BTLR short for **B**ayesian **T**ransparent **L**ine **R**emover.

Chapter 5 Infrared Analysis

This chapter shows experiments used to evaluate our blotch (BTBR) and line (BTLR) removal techniques. Comparisons with ground truth estimates show that our restoration techniques generate good removal despite motion and texture complexities. Here ground-truth estimates are generated by transforming IR scans of corruptions from their original grayscale domain to the corruption opacity domain. This generates ground-truth corruption opacities which are used to weight out the effect of the corruptions and hence unveil the underlying original data.

Chapter 6: Reflection Detection in Images Sequences

This chapter proposes an approach for automated detection of reflections in natural image sequences. This is done by analyzing feature point trajectories. A set of weak detectors is generated. A strong detector is proposed by combining those weak detectors. We generate a solution within a Machine Learning framework, use spatial and temporal information and results show high reflection detection rate with rejection to regions of complicated motions. The proposed detection technique is called FEAPARD, short for **Fea**ture **P**oint **A**nalyses for **R**eflection **D**etection.

Chapter 7: Motion Estimation for Regions of Reflection

This chapter proposes an approach for multiple motion estimation for regions of reflections. It models the correct motions as the ones generating the best foreground and background reconstructions of the examined reflection. Feature point trajectories act as a pool of candidates for the motions. We generate a solution within a Bayesian framework, use novel temporal priors and results shows good motion estimation despite motion inconsistencies. We show how to weight out erroneous motion estimates in regions not containing reflections. This is done using the detection masks generated from FEAPARD, our reflection detection technique. The proposed multiple motion estimator is called BIMS, short for **B**ayesian **I**nference for **M**ultiple motion estimation through layer **S**eparation.

Chapter 8: Conclusions

The final chapter assesses the contributions of this thesis and outlines some directions for future work.

1.2 Contributions of this thesis

Aspects of novelty in this thesis can be summarized as follows

- Incorporation of a model for semi-transparency into a Bayesian framework for blotch removal.

- A new analysis of corruption IR scans to create accurate ground truth for corruption removal.
- Hybrid schemes for solution of the Bayesian corruption removal problem.
- A new technique for line scratch removal based on recursive filtering using the same Bayesian framework of blotch removal.
- A technique for decomposing a color still image containing reflection into two images containing the structures of the source layers. We do not claim that this technique could be used to fully remove reflections from videos. What we claim is that the extracted layers can be useful for reflection detection since on a block basis, reflection is reduced.
- Diagnostic tools for reflection detection based on analyzing feature point trajectories. This generates a set of weak reflection detectors.
- A scheme for combining the weak detectors in one strong reflection detector using Adaboost.
- The generation of dense reflection detection maps from sparse detections and using thresholding by hysteresis to avoid selecting particular thresholds for the system parameters.
- Incorporating spatio-temporal information which reject spatially and temporally impulsive reflection detections.
- A technique for multiple motion estimation for regions of reflections through layer separation.
- Incorporation of novel temporal priors.
- Using reflection masks to mask out erroneous motion estimates in regions not containing reflections. These masks are generated from our reflection detection technique.

1.3 Publications

Portions of the work described in this thesis have appeared in the following publications:

- "Extraction of Non-Binary Blotch Mattes" by Mohamed Ahmed, Francois Pitie, and Anil Kokaram, in *Proceedings of the International Conference on Image Processing (ICIP '09)*, pages 2757-2760, Cairo, Egypt, November 2009.
- "Reflection Detection in Image Sequences" by Mohamed Ahmed, Francois Pitie, and Anil Kokaram, in *Proceedings of IEEE Computer Vision and Pattern Recognition (CVPR '11)*, pages 705-712, Colorado, U.S.A, June 2011.

- "Motion Estimation for Regions of Reflections Through Layer Separation" by Mohamed Ahmed, Francois Pitie, and Anil Kokaram, *To appear in the IEEE European Conference on Visual Media Production (CVMP '11)*, London, UK, November 2011.

- "Blotch and Scratch Removal in Archived Film Using a Semi-transparent Corruption Model" by Mohamed Ahmed, Francois Pitie, and Anil Kokaram, *Under review in the International Journal of Computer Vision (IJCV).*

2

Handling Transparency in Natural Images: Related work

Following from the introduction to this work, we seek to model transparency as a mixture of two layers. The essential statement is repeated here for clarity as follows. The observed image mixture M is modeled as a linear combination between the foreground layer F and the background layer B as follows.

$$M(\mathbf{x}) = \alpha(\mathbf{x})F(\mathbf{x}) + (\mathbf{1} - \alpha(\mathbf{x}))\mathbf{B}(\mathbf{x}) \tag{2.1}$$

Here $\mathbf{x}$ denote the pels of the considered image while α is a parameter that measures the blend between the foreground and background layers. α is called 'opacity' in the literature. $\alpha(\mathbf{x}) = 1$ represents complete obliteration of the background layer at pel $\mathbf{x}$ while $\alpha(\mathbf{x}) = 0$ represents complete absence of the foreground layer at the examined site (see figure 2.1).

The important observation is that equation 2.1 is the same as the well known compositing equation used for extracting object mattes (see figure 2.2, regions shown in red) as in [8,9,17,30, 31,33,36,52,58,62,64–66,70,76,82,90,91,96]. We will draw on this body of work in designing our own algorithms so the first part of this review introduces the important ideas we require from that literature. Note however in this thesis we do not address the problem of matte extraction in an explicit form. Instead we just use ideas from the matting problem. Hence in this chapter we do not present an overview of the matting techniques. However, we focus on discussing the matting technique of Chuang et al. [17] as it is used extensively in the corruption removal technique proposed in this thesis. We use the technique of Chuang et al. as it is the most

Figure 2.1: *Mixing the image of lenna (bottom left) with an image of a crowd of people (top left) using equation 2.1 with different opacity values (α). Here the crowd of people is the foreground while lenna is the background. Opacity values used (clockwise from top left) are* $\alpha = [1, 0.85, 0.5, 0.35, 0]$. *$\alpha$ is fixed over an entire mixture. $\alpha = 1$ shows complete obliteration of the background (lenna) while $\alpha = 0$ shows complete absence of the foreground (the crowd)*

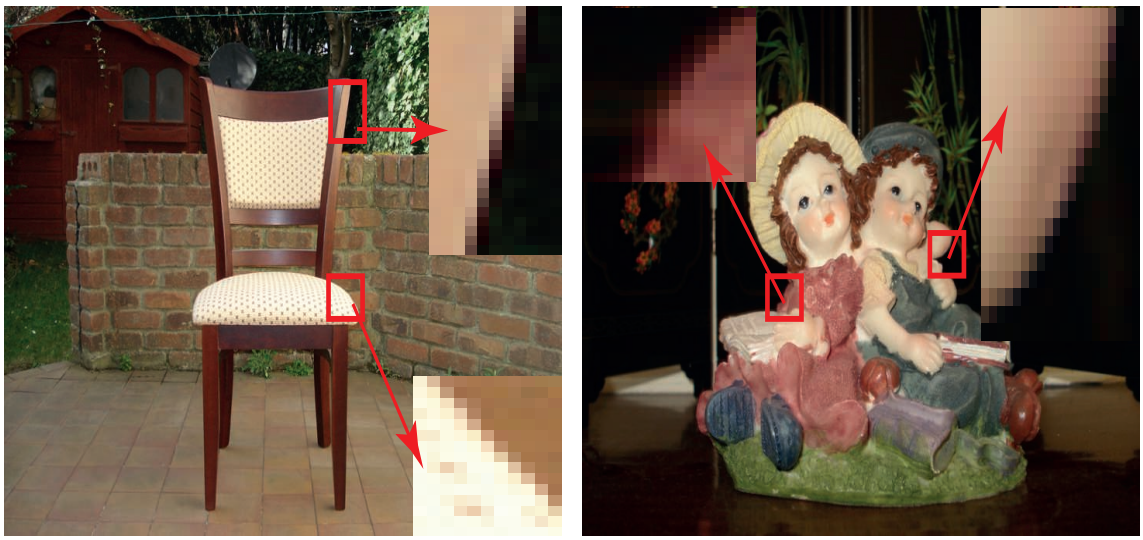


Figure 2.2: *Transparency occurs around object boundaries (examples shown in red) where the observed color is a mixture between the foreground and background object. The foreground object for the left and right images are the chair and the toy respectively. The soft blend between the foreground and background objects (shown in red) is called 'Matte'.*

popular matting technique that uses a Bayesian framework. The Bayesian framework is the most general framework for solutions. This allows the Chuang et al. approach to be extended to other problems.

There is no previous work on addressing reflection detection. However many work exists on layer separation from natural image mixtures [12, 16, 22, 26, 39–42, 45, 53, 54, 71, 73, 74, 77, 94, 95, 97, 98]. Again, we draw on that body of work to develop our reflection detection and multiple motion estimation ideas and so the next part of the review considers these. However as this thesis does not address the problem of layer separation in an explicit form, we focus more on separation techniques that are used in the main contributions of this thesis.

Finally, several authors have considered the problem of multiple motion estimation, although not many have attempted to assign the motion to the correct image layers [3,7,59,61,75,81,86,88]. Hence the final part of this chapter presents a review of current multiple motion estimators.

This chapter starts by a quick review of the matte extraction problem. We do not present a literature review of the matting techniques. Instead, we mainly discuss the Chuang et al. [17] approach as it is the only matting technique used in our contributions. The chapter then goes on to discuss techniques for reflection separation. Here we discuss techniques designed for still images as well as techniques designed for image sequences. Again, we mainly focus on separation techniques that are used in our contributions. Last, previous techniques for estimating the foreground and background motions in regions of reflections are discussed.

2.1 Matte Extraction

Plenty of techniques exist for matte extraction [9, 17, 33, 52, 58, 62, 64, 70, 76, 82, 91]. Those techniques take two main different approaches; 1) deterministic [9,52,58,62,76,82] and 2) probabilistic [17, 33, 64, 70, 91]. Among the techniques that take probabilistic approaches, Chuang et al. [17] were the first to formulate the matting problem within a Bayesian framework. The Bayesian framework is the most general framework for solutions. This allows [17] to be extended to other problems. In addition, more recent matting techniques that use a Bayesian framework build on similar ideas from [17]. Due to those reasons we extensively use the technique of Chuang et al. in our corruption removal techniques. Their technique is commonly known as 'Bayesian Matting'.

Bayesian Matting requires an initial segmentation of the observed image into three main regions being: 1) definite foreground 2) definite background and 3) unknown. This segmentation is called a 'Trimap' and can be manually supplied. Bayesian Matting then solves for the missing parameters $(\alpha(\mathbf{x}), B(\mathbf{x}), F(\mathbf{x}))$ for every pel $\mathbf{x}$ in the unknown region using information from the nearby definite foreground and background regions. We discuss this technique in more detail next.

2.1.1 A Bayesian Approach for Digital Image Matting

The Chuang et al. approach [17] derives an estimate for the (α, F, B) for every pel $\mathbf{x}$ in the unknown region from the posterior $P(\alpha, F, B|M)$ (where $\mathbf{x}$ is dropped for clarity). The posterior is factorized in a Bayesian fashion as follows

$$P(\alpha, F, B|M) = P(M|\alpha, F, B)P(B)P(F) \tag{2.2}$$

The likelihood $P(M|\alpha, F, B)$ ensures that the estimated (α, F, B) generate the observed mixture M while the priors $P(B)$ and $P(F)$ enforces the estimated B and F to be close to the nearby definite background-foreground data. This approach does not impose any prior information on α.

2.1.1.1 Modeling the priors $P(F)$ and $P(B)$

For every pel in the unknown region, the foreground prior $P(F)$ is modeled as a mixture of Gaussians. That mixture of Gaussians is estimated from the nearby definite foreground data. This forces the reconstructed foreground data to be consistent with the surrounding pels. Foreground samples are collected by extending a circular patch from the examined site until a minimum number of definite foreground pels is included. The patch is then segmented into $\mathcal{M}_c$ color clusters using a color quantization algorithm [60]. This yields a mixture of $\mathcal{M}_c$ Gaussians, each with mean and covariance $(\bar{\mathbf{F}}, \mathbf{R}_F)$. The background prior $P(B)$ is also modeled in a similar way as a mixture of $\mathcal{M}_c$ Gaussians using samples from the nearby definite background region. Note that for simplicity we used the same number of foreground and background color clusters.

2.1.1.2 Solving for (α, F, B)

Given observation noise $\mathcal{N}(0, \sigma_e^2)$ in the compositing model of equation. 2.1 and substituting $P(F)$ and $P(B)$ in equation 2.2, the Posterior for (α, F, B) for the jth background-foreground color cluster pair is as follows

$$P(\alpha, F, B|M) \propto$$
$$\exp - \left(\frac{||M - \alpha F - (1-\alpha)B||^2}{2\sigma_e^2} + (F - \bar{\mathbf{F}}_j)^T \mathbf{R}_{F_j} (F - \bar{\mathbf{F}}_j) + (B - \bar{\mathbf{B}}_j)^T \mathbf{R}_{B_j} (B - \bar{\mathbf{B}}_j) \right) \quad (2.3)$$

Unfortunately, the priors for F and B are Gaussian mixtures and this makes it difficult to propose a closed form solution. Instead, given $\mathcal{M}_c$ Gaussians in each cluster, a candidate solution is generated for each possible cluster pair. There are $\mathcal{M}_c^2$ possible cluster pairs, hence this yields $\mathcal{M}_c^2$ candidate solutions. Since each candidate solution is generated from only a pair of Gaussians, the solution for each candidate is much more tractable. Then, having generated the $\mathcal{M}_c^2$ candidates, the set that maximizes the posterior above is selected as the required MAP estimate.

The candidate generation step iterates between two closed form estimates for (F, B) and α. Appendix A describes in more detail the process, but for the jth background-foreground cluster pair, the system of equations yielding the estimates is as follows.

$$\begin{bmatrix} \sigma_e^2 \mathbf{R}_{F_j}^{-1} + I\alpha_j^2 & I\alpha_j(1-\alpha_j) \\ I\alpha_j(1-\alpha_j) & \sigma_e^2 \mathbf{R}_{B_j}^{-1} + I(1-\alpha_j)^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \sigma_e^2 \mathbf{R}_{F_j}^{-1} \bar{F}_j + \alpha_j M \\ \sigma_e^2 \mathbf{R}_{B_j}^{-1} \bar{B}_j + (1-\alpha_j) M \end{bmatrix} \quad (2.4)$$

$$\alpha_j = \frac{(M - B_j)^T (F - B_j)}{||F - B_j||^2} \quad (2.5)$$

Here $\mathbf{I}$ is the 3×3 Identity matrix. The optimal (α_j, F_j, B_j) estimates for the jth background-foreground color cluster pair is calculated by iterating between equation 2.4 and equation 2.5. The initial value for α is set to 0.5 and subsequent initial estimates for α use the mean of previously calculated opacities.

Figure 2.3 shows the extracted matte for an image of a girl using this approach (our implementation). The soft blend between the foreground and background at the borders of the girl's hair is shown by the estimated non-binary opacity values. The estimated opacities are varying smoothly from $\alpha = 0$ near the background to $\alpha = 1$ near the foreground.

2.2 Reflection Separation

Techniques for estimating the foreground and background layers of reflections largely fall into two main categories. The first category is designed to separate mixtures of still images [12,16,22, 26,39–42,45,53,54,71,73,77,95,97,98] while the second category is designed to handle reflections in image sequences [74, 94]. All separation techniques estimate the underlying layers up to a scale. Most still image techniques require two mixtures of the same foreground and background

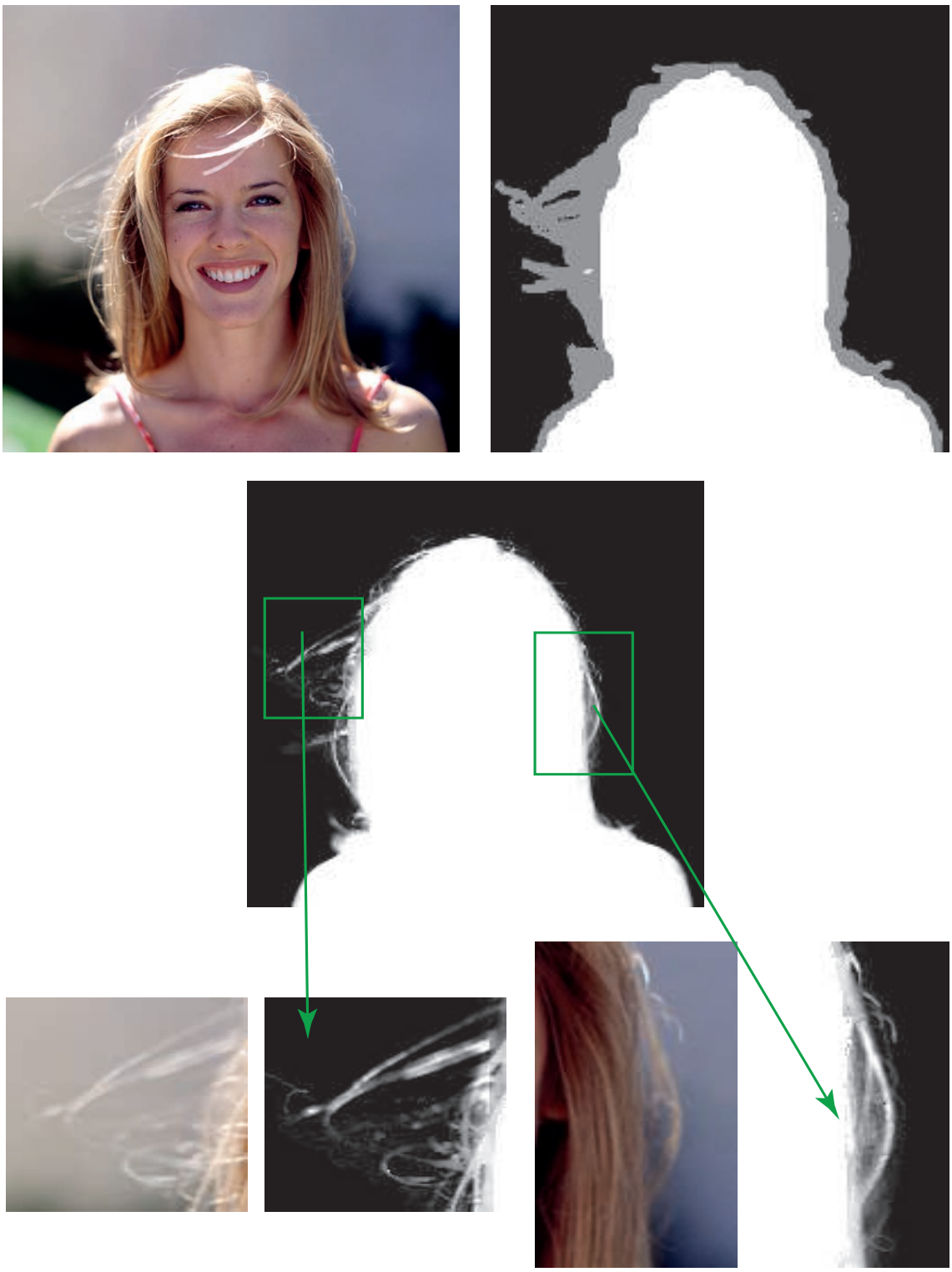


Figure 2.3: *Top: An image for a girl (from Chuang et al. [17]) and the corresponding trimap (right). White, grey and black colors in the trimap represent definite foreground, unknown, and definite background respectively. Here the girl is treated as the foreground object and scene behind the girl is the background. Middle: Extracted matte using Chuang et al. Bayesian Matting (our implementation). Bottom: Areas shown in green (in the second row) zoomed on with the original image shown in left and the extracted opacities shown in right. The soft blend between the foreground and background objects is shown by the estimated non-binary opacity values.*

layers under different mixing conditions. The two compositing equations for the two mixtures M_1 and M_2 can be written as

$$\begin{aligned} M_1(\mathbf{x}) &= \alpha_1 F(\mathbf{x}) + (1-\alpha_1)B(\mathbf{x}) \\ M_2(\mathbf{x}) &= \alpha_2 F(\mathbf{x}) + (1-\alpha_2)B(\mathbf{x}) \end{aligned} \tag{2.6}$$

Here $\mathbf{x}$ denote the pels in the examined mixtures where $[\alpha_1, \alpha_2]$ are the mixing parameters (see figure 2.4). For layer separation to be achieved each layer should have definite structure (e.g. none of the layers should be devoid of texture) and both layer structures should be different from each other. The absence of the first assumption could make the observed mixture look like a scaled version of the layer that has distinctive structure. This will cause separation techniques to interpret the observed mixture as being just one layer since all separation techniques recover layers up to a scale. The absence of the second assumption could make layers look similar and hence hard to separate from each other.

Most separation techniques for still images solve for the underlying layers F and B in a way such that the structural similarity between the estimated layers is minimized. Different still image separation techniques use different approaches to impose structure dissimilarity between the separated layers. Sarel et al. [73, 74] defined the structural similarity as the similarity in the grayscale correlation of the separated layers. They introduced a measure named GNGC (Generalized Normalized Grayscale Correlation) that measures the grayscale correlation between the layers. This measure is robust to grayscale deformations and generates good separation results [73]. Several authors imposed structural dissimilarity between the separate layers by maximizing the sparseness of edges and corners in the separated layers [12, 22, 45, 53, 77, 97]. This is done by removing the edges/corners belonging to B from the estimated F layer, and similarly removing the edges/corners belonging to F from the estimated B layer. Hence, F and B are solved in a way such that the edges/corners do not occur in the same spatial location in both layers i.e. separated layers have large structural dissimilarity.

Techniques for reflection separation for image sequences do not require two mixtures of the same foreground and background layers. Instead, they require each one of the foreground and background layers to be undergoing different illumination variations through time [74, 94]. Here separation is done separately for each layer. This is done by temporally integrating frames.

In the remaining part of this section we discuss layer separation techniques in more detail. For still images we discuss separation using the GNGC measure of Sarel et al. [73] and by maximizing sparseness of edges and corners in the separated layers. We then go on to address layer separation for image sequences in more detail.

2.2.1 Separation Using GNGC

Sarel et al. [73] proposed an approach to estimate the foreground and background layers F and B from two mixtures M_1 and M_2 by minimizing the grayscale correlation between the separated

Figure 2.4: *Two layers (top) mixed together to generate two different mixtures (bottom). Here Lenna is the foreground layer while Oranges is the background. Mixing parameters used for the mixtures (in the bottom) are, from Left,* $[\alpha_1, \alpha_2] = [0.5, 0.4]$. *Most still image separation techniques seek to estimate the original layers (top) given at least two mixtures of the same foreground and background layers (bottom).*

layers. They introduced a new measure named Generalized Normalized Gray-scale Correlation (GNGC) for measuring the structural correlation between the separated layers. GNGC for two layers F and B is defined as follows

$$GNGC(F,B) = \frac{\sum_{\mathbf{x}=1}^{k} C_{\mathbf{x}}^2(F,B)}{\sum_{\mathbf{x}=1}^{k} V_{\mathbf{x}}(F).V_{\mathbf{x}}(B)} \tag{2.7}$$

Here $\mathbf{x}$ indexes locations of the pels of the examined layers, while C_x and V_x are the covariance and variance of a small image patch (typically 7×7) centered on the examined pel. Sarel et al. [73] showed that GNGC is robust to spatially variant grayscale deformations on the examined layers. This property makes GNGC suitable for layer separation in noisy measurements.

2.2.1.1 Information Layer Exchange

Assuming that F is more dominant than B in M_1 and B is more dominant than F in M_2, Sarel et al. [73] proposed an approach for estimating the underling layers F and B named 'Information Layer Exchange'. The idea here is to exchange information between the two observed mixtures M_1 and M_2 till the component of one of the layer (say B) disappears from one of the mixtures (say M_1) leaving behind the other layer (F in this case). Similarly, B can be estimated by removing the component of F from M_2.

Given equation 2.6 as the mixing model for the observed mixtures M_1 and M_2, there must exist a scalar γ_1 such that $\hat{F} = M_1 - \gamma_1 M_2$ contains only the structure of F (here $\hat{F}$ is the estimated F). Hence the optimal value of γ_1 is the one that minimizes the structural similarity between $\hat{F}$ and M_2. Using GNGC to measure the structural similarity, the optimal estimate of γ_1 is

$$\gamma_1 = \arg\min(GNGC(M_2, M_1 - \gamma_1 M_2)) \tag{2.8}$$

γ_1 is estimated through an exhaustive search (typically in the range of $[-1 : 0.1 : 1]$). Given the estimated value of γ_1, the foreground layer is recovered (up to a scale) using $\hat{F} = M_1 - \gamma_1 M_2$. Now $\hat{B}$ can be estimated by removing the foreground component from M_2. Hence $\hat{B}$ is expressed as $\hat{B} = M_2 - \gamma_2 \hat{F}$ where the optimal value of γ_2 minimizes the structural similarity between $\hat{B}$ and $\hat{F}$. γ_2 is estimated through an exhaustive search by minimizing $GNGC(\hat{F}, M_2 - \gamma \hat{F})$.

Sarel et al. [73] proposed to repeat the above optimization process a few times to obtain a cleaner layer separation. At each iteration, the previously recovered layer $\hat{F}$ and $\hat{B}$ serve as the new mixtures as follows

$$\hat{F}^{k+1} = \hat{F}^k - {\gamma_1}^k \hat{B}^k \tag{2.9}$$

$$\hat{B}^{k+1} = \hat{B}^k - {\gamma_2}^k \hat{F}^{k+1} \tag{2.10}$$

where k is the iteration number.

Figure 2.5 shows the result of estimating Lenna and Oranges of figure 2.4 (bottom) using this approach. Here three iterations (k=2) are used. In the regions shown in red the remains

of Lenna are reduced as more iterations are used. In comparison with the original mixtures (figure 2.4, bottom), Oranges are removed significantly from the reconstructed Lenna (left) and Lenna is removed significantly from the reconstructed Oranges (right). However, GNGC failed to generate complete layer separation mainly in the reconstructed Lenna where some remains of the Oranges still exist (shown in green, bottom). This is known as the 'Ghost Effect'. The reason for this effect goes back to the first iteration of this approach in estimating Lenna (see top, left). In the first iteration the ghost effect of the Oranges helped in minimizing the GNGC between the estimated Lenna (at this iteration) with M_2. Hence, this reconstruction of Lenna was favored over the rest. This reconstruction error propagated through the remaining iterations.

2.2.2 Maximizing Layers Sparseness

Several authors have shown successful layer separation by maximizing the sparseness of edges and corners in the estimated layers [12, 22, 45, 53, 77, 97]. This is done by solving for F and B in a way such that edges or corners do not occur in the same spatial location in the reconstructed foreground and background layers. This imposes structural dissimilarity on the separated layers. Diamantaras et al. [22] was the first to show that layers can be estimated up to a scale if the examined layers are sparse in the sense that they have zero mean [22]. Bronstein et al. [12] showed that image edges can generate such sparse representations for the layers. Later Souidene et al. [77] generated successful separation by explicitly maximizing the sparseness of the edges in the separated layers. Next we introduce the ideas of Diamantaras et al. [22] before going on to illustrate the use of edges and corners as sparse image features for layer separation.

2.2.2.1 Diamantaras Separation Using Image Sparseness

Denote $\bar{\mathbf{F}}(\mathbf{x}) = \alpha_1 F(\mathbf{x})$ and $\bar{\mathbf{B}}(\mathbf{x}) = (1-\alpha_1)B(\mathbf{x})$, then the ratio of the image mixtures $r(\mathbf{x})$ is as follows

$$r(\mathbf{x}) = \frac{M_2(\mathbf{x})}{M_1(\mathbf{x})} = \frac{a_1 + a_2 r_y(\mathbf{x})}{1 + r_y(\mathbf{x})} \tag{2.11}$$

where $r_y(\mathbf{x}) = \bar{\mathbf{B}}(\mathbf{x})/\bar{\mathbf{F}}(\mathbf{x})$, $a_1 = \alpha_2/\alpha_1$, $a_2 = (1-\alpha_2)/(1-\alpha_1)$. If a_1 and a_2 can be measured, the underlying layers F and B can be estimated up to a scale as follows

$$\begin{bmatrix} \bar{\mathbf{F}}(\mathbf{x}) \\ \bar{\mathbf{B}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ a_1 & a_2 \end{bmatrix}^{-1} \begin{bmatrix} M_1(\mathbf{x}) \\ M_2(\mathbf{x}) \end{bmatrix} \tag{2.12}$$

where $\bar{\mathbf{F}}$ and $\bar{\mathbf{B}}$ are the scaled versions of F and B respectively. a_1 and a_2 can be estimated at sites where $r_y(\mathbf{x}) = 0$ and $r_y(\mathbf{x}) = \infty$ as follows

$$r(\mathbf{x})|_{r_y(\mathbf{x})=0} = a_1 \tag{2.13}$$

$$r(\mathbf{x})|_{r_y(\mathbf{x})=\infty} = a_2 \tag{2.14}$$

$r_y(\mathbf{x}) = 0$ implies that at site $\mathbf{x}$ $F(\mathbf{x}) \neq 0$ and $B(\mathbf{x}) = 0$ while $r_y(\mathbf{x}) = \infty$ implies that at $\mathbf{x}$ $F(\mathbf{x}) = 0$ and $B(\mathbf{x}) \neq 0$. Hence, Diamantaras et al. states that one can estimate the mixing

Figure 2.5: *Estimating the source layers of the mixtures in figure 2.4 (bottom) using Layer Exchange Information with GNGC [74]. Pictures are scaled by a factor of 7 (left column) and 1.25 (right column) for illustration clarity. Each row represents one iteration of the process with the extracted foreground and background layer on the left and right respectively. Calculated mixing parameters* $[\gamma_1, \gamma_2]$ *are (from top)* $[0.9, 1]$, $[0, 0.75]$ *and* $[0, 0.2]$. *In comparison with the original mixtures in figure 2.4 (bottom), this approach was able to reduce the mixing effect in the final reconstructed layers (bottom). In the regions shown in red the remains of Lenna are reduced as more iterations are used. However, some parts of the background Oranges still remain in the reconstructed foreground Lenna (mainly shown in green).*

parameters a_1 and a_2 and consequently the scaled versions of F and B if there at least one pel in the observed mixtures where F vanishes and B remains ($r_y(\mathbf{x}) = \infty$), and another pel where B vanishes and F remains ($r_y(\mathbf{x}) = 0$). These constraints are common if the examined layers are processed in a zero-mean sparse representation instead of the original grayscale representation. The image first order derivative is an example of a zero-mean sparse representation (see figure 2.6 top)

2.2.2.2 Layer Separation by Maximizing Edges Sparseness

Bronstein et al. [12] proposed the use of image edges as a zero-mean sparse representation for the examined mixtures/layers. Sparse representations of the observed mixtures are generated by applying the first order derivative T on M_1 and M_2 as follows

$$\begin{aligned} T(M_1(\mathbf{x})) &= T(\alpha_1 F(\mathbf{x}) + (1-\alpha_1)B(\mathbf{x})) \\ T(M_2(\mathbf{x})) &= T(\alpha_2 F(\mathbf{x}) + (1-\alpha_2)B(\mathbf{x})) \end{aligned} \tag{2.15}$$

Due to the linearity of T one can rewrite equation 2.15 as follows

$$\begin{aligned} M_1^d(\mathbf{x}) &= \alpha_1 F^d(\mathbf{x}) + (1-\alpha_1)B^d(\mathbf{x}) \\ M_2^d(\mathbf{x}) &= \alpha_2 F^d(\mathbf{x}) + (1-\alpha_2)B^d(\mathbf{x}) \end{aligned} \tag{2.16}$$

where $[M_1^d, M_2^d, F^d, B^d]$ are the first order derivatives of $[M_1, M_2, F, B]$ respectively. The mixing parameters a_1 and a_2 of Diamantaras et al. [22] can now be estimated using the same approach of Diamantaras but with processing $[M_1^d, M_2^d, F^d, B^d]$ instead of $[M_1, M_2, F, B]$. That is

$$r^d(\mathbf{x})|_{r_y^d(\mathbf{x})=0} = a_1 \tag{2.17}$$

$$r^d(\mathbf{x})|_{r_y^d(\mathbf{x})=\infty} = a_2 \tag{2.18}$$

where $r^d = M_2^d / M_1^d$ and $r_y^d = \frac{(1-\alpha_1)B^d}{\alpha_1 F^d}$. This approach imposes structural dissimilarity on the separated layers in an implicit form as it assumes edges are not present in the same spatial location in both separated layers.

Plotting M_2^d versus M_1^d generates two lines, with slopes a_1 and a_2 (see figure 2.6, (Bottom, left)). Bronstein et al. [12] estimated the two dominant slopes $[a_1, a_2]$ by looking for the two highest peaks in the mixtures ratio histogram of every point in M_2^d versus M_1^d (see figure 2.6 bottom, right). Once a_1 and a_2 are estimated, scaled versions of F and B are calculated using equation 2.12.

Figure 2.6 shows the process of estimating Lenna and Oranges of figure 2.4 (bottom) using this approach of Bronstein et al. Recall the mixing parameters used to generate those mixtures are $\alpha_1 = 0.5$ and $\alpha_2 = 0.4$. Hence, the ground truth estimates of a_1 and a_2 are 0.8 and 1.2 respectively. Figure 2.6 (bottom left) shows the plot of M_2^d versus M_1^d for the mixtures of Lenna and Oranges in figure 2.6 (top). Most points in this plot lie along two lines as expected.

Figure 2.6 (bottom right) shows the process of extracting a_1 and a_2 from the left graph. This is done by first estimating the mixture ratio M_2^d/M_1^d for every point on the left graph and then finding the two most common ratios. Figure 2.6 (bottom right) shows the histogram of all mixture ratios for the left plot. The two peaks here correspond to the two dominant mixing ratios i.e. a_1 and a_2. Estimated values for the mixing rations are $a_1 = 0.8$ and $a_2 = 1.2$. The estimated values of a_1 and a_2 match the ground-truth estimates. Figure 2.7 shows the reconstructed Lenna and Oranges using the estimated mixing parameters. As shown, both layers are well reconstructed.

Later, Souidene et al. [77] used a similar layer separation approach to Bronestein et al. [12] by explicitly maximizing the sparseness of the edges in the separated layers. Assuming that F is more dominant than B in the observed original mixture M_1, F is estimated by removing the edges that correspond to B from M_1. This is done by minimizing the strength of edges in M_1. Similarly, assuming that B is more dominant than F in the observed original mixture M_2, B is estimated by removing the edges that correspond to F from M_2. Hence, given a set of F and B candidates, the optimal F and B pair is selected as the one with minimum total strength of image gradients i.e. edges in both layers are made more sparse.

2.2.2.3 Layer Separation by Maximizing Corner Sparseness

Levin et al. [54] proposed a separation approach that is similar to Souidene et al. [77] in the sense that they both maximize the sparseness of specific image features in the separated layers. However, instead of using edges for sparse features as in Souidene et al. [77], Levin et al. [54] uses image corners. Levin et al. noted that due to the sparse nature of corners, the correct estimate of the foreground and background layers would often have the minimum total number of corners among other possible foreground-background pairs. Hence the correct F and B estimates are the ones that maximize the sparseness of corners in the separated layer. Based on this observation they proposed a separation technique that can estimate the foreground and background layers given just one image mixture [54]. The mixture is processed in small blocks. For each block, different foreground-background decompositions of M are selected from a database of natural images. The foreground-background layer combination with minimum total number of corners is selected as the correct estimate of the underling layers.

Figure 2.8 shows the reconstructed Lenna and Oranges of figure 2.4 (bottom) using Soudiene et al. [77] and Levin et al. [54]. We are more interested in examining the ability of performing layer separation though minimizing the sparseness of edges and corners in the separated layers rather than examining the optimization methods proposed by those authors to estimate the optimal mixing parameters. Therefore we use Sarel et al. [73] Layer Information Exchange as an optimization scheme to find the optimal mixing parameters. For Soudien et al. F and B are solved in a way such that the sum of strength of edges in the separated layers is minimized. Here edges are extracted using a simple first order horizontal derivative. For Levin et al. F and B are

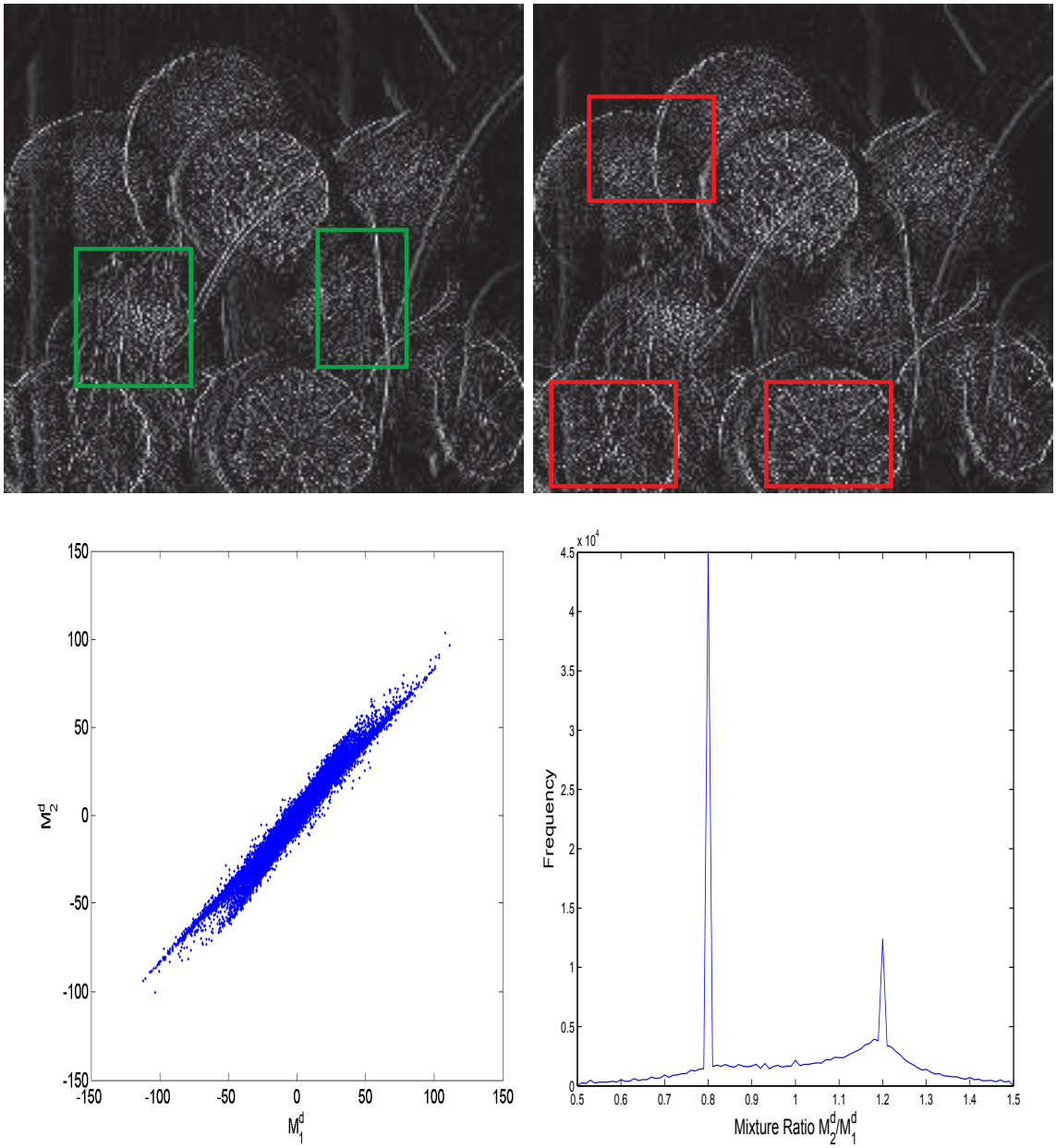


Figure 2.6: *Top: Horizontal first order derivative for the mixtures in figure 2.4 (bottom). Edges hardly occur in the same spatial location in both Lenna and Oranges. For instance, the edges shown in green belong mainly to Lenna's hair and hat while the edges shown in red are mainly the Oranges'. Bottom (left): The plot of* M_2^d *(top right) versus* M_1^d *(top left). Most points lie across two lines. The slopes of those two lines can be calculated using* M_2^d/M_1^d *and correspond to the mixing parameters* a_1 *and* a_2*. Bottom (right): The histogram of the mixture ratio* M_2^d/M_1^d *for every point on the left graph. The two highest peaks in the histogram correspond to the two dominant mixture ratios* M_2^d/M_1^d *in the left plot i.e. correspond to* a_1 *and* a_2*.*

Figure 2.7: *Reconstructed Lenna and Oranges of figure 2.4 (bottom) using Bronestein et al. separation approach. Pictures are scaled by a factor of 2 for illustration clarity. Results show good reconstruction of Lenna and Oranges.*

solved in a way such that the sum of corners in the separated layers is minimized. Here Harris corners detector [35] is used to extract the corners [35]. Both approaches of Soudiene et al. and Levin et al. generate the same reconstruction for Lenna and Oranges (see figure 2.8). However the layers are much better separated from each other than when using GNGC (see figure 2.5, bottom). The 'Ghost Effect' generated by GNGC (shown in green in figure 2.5, bottom) is reduced significantly in figure 2.8.

2.2.3 Layer Separation For Image Sequences

There is relatively little published work that considers separating reflections in image sequences [74, 94], however Weiss' technique [94] is the most commonly cited. Although we are not interested in high quality layer separation in this thesis, the models employed in that work are of course relevant to transparency handling in general. Hence we consider that background material as part of this review. Weiss et al [94] assume that one of the two layers is moving, hence the moving layer contains luminance variations along the non-motion compensated direction because of the movement. Therefore, filtering in some way along the zero motion trajectory will enhance the static layer and suppress the moving layer.

Weiss articulates these ideas in a probabilistic fashion by first assuming that the priors for the foreground (moving layer) gradients $[F_x, F_y]$ are Laplacian (with parameter ρ) as follows.

$$P(F_x^i) \propto \exp^{-\rho|F_x^i|} \quad ; \quad P(F_y^i) \propto \exp^{-\rho|F_y^i|} \tag{2.19}$$

where F_x^i is the gradient of the foreground layer in the horizontal direction in frame i (and similarly for F_y^i but gradients taken in vertical direction). In his work, Weiss proposes that

Figure 2.8: *Reconstructed Lenna and Oranges of figure 2.4 (bottom) using Soudiene et al. [77] separation approach (our implementation). Here we used Sarel et al. Layer Exchange Information optimization scheme however we solve for F and B in a way such that the sum of gradients in the separated layers is minimized. Pictures are scaled by a factor of 2.5 (left column) and 2 (right column) for illustration clarity. The estimated mixing parameters $[\gamma_1, \gamma_2]$ (see equation 2.8) are* 0.75 *and* 0.8 *for Lenna and Oranges respectively. Results show better separation of Lenna and Oranges over using GNGC as in figure 2.5 (Bottom). Here we do not have a strong ghost effect as the one in figure 2.5 (bottom, shown in green). The same reconstruction of Lenna and Oranges is generated using Levin et al. [54] separation approach (our implementation). Here we use Sarel et al. Layer Exchange Information optimization scheme however we solve for F and B in a way such that the sum of the corners in the separated layers is minimized.*

$M = F + B$. However, if we assume that we are only interested in scaled versions of F, B then $M = \alpha F + (1-\alpha)B$ in our model becomes $M = F' + B'$. Hence the Likelihood w.r.t. B' across T frames is then

$$P(M|B_x^i, B_y^i) \propto \exp\left\{-\rho \sum_{i=1}^{T} |M_x^i - B_x| + |M_y^i - B_y|\right\} \tag{2.20}$$

Therefore the ML estimate for the gradients of the background layer are given by the median of the samples along T frames in the observed image M. Given those gradients, the actual background layer can be recovered by inverting the gradient operation in the usual way [94].

This process showed interesting results on separating a static background from reflections in image sequences. Sarel et al. [74] in 2005, extended the Weiss approach to separate moving background layers. The idea here is to temporally align the background layer over a window of frames. This however requires estimating the motion of the background layer, a problem that is not straightforward for regions in reflections. Hence, Sarel et. al [74] restricted attention to

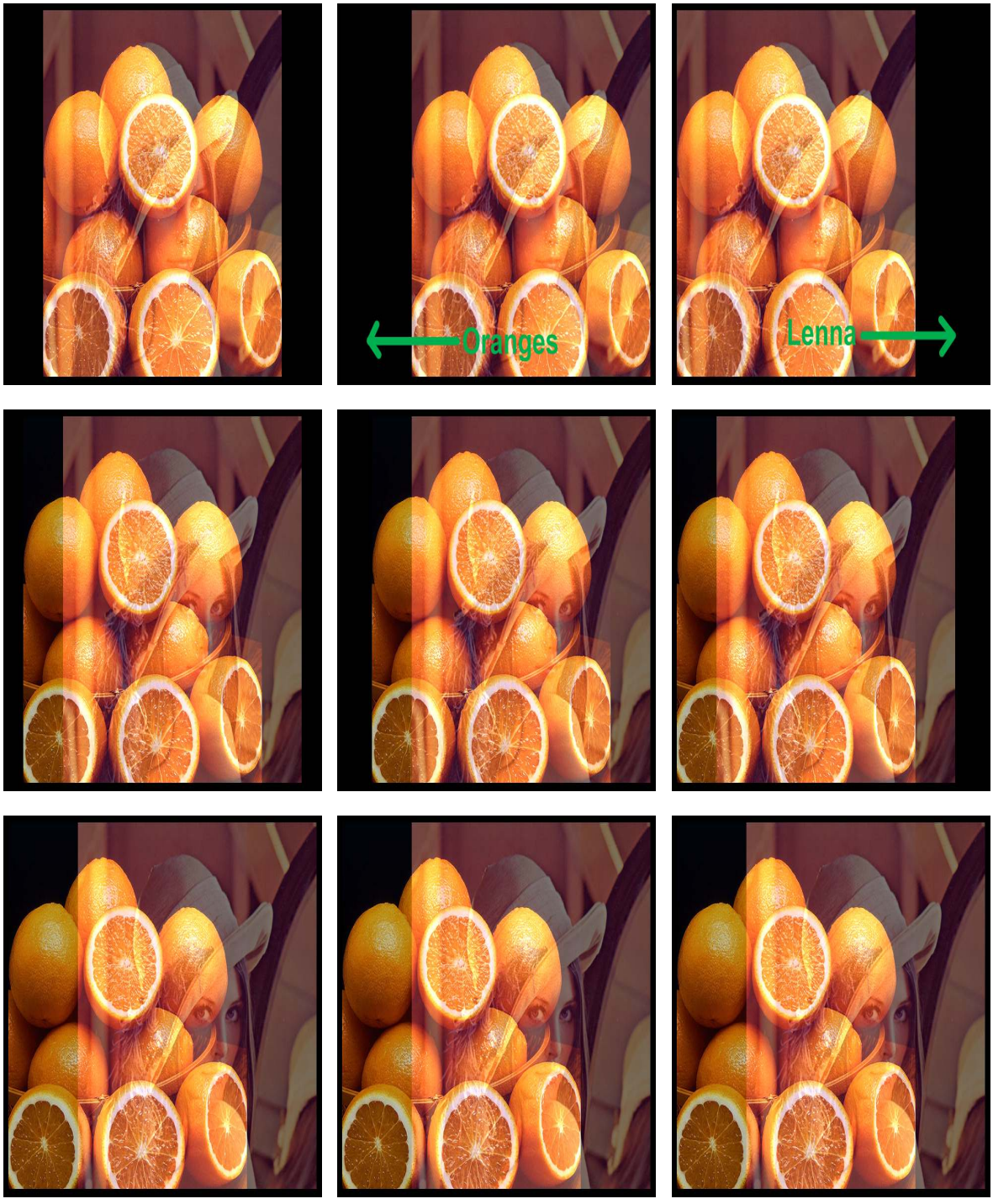


Figure 2.9: *Left, from top: Frames 1, 5 and 8 from a ten frame image sequence created by mixing Lenna with Oranges with* $\alpha = 0.5$. *The motion of both layers is constant through the entire sequence and it is* $[0, 10]$ *and* $[0, -10]$ *for Lenna and Oranges respectively (shown in green in the first row). To reconstruct Lenna and Oranges, the sequences must be stabilized over those two layers separately. In the middle column the sequence is stabilized over Lenna while in the third column the sequence is stabilized over Oranges.*

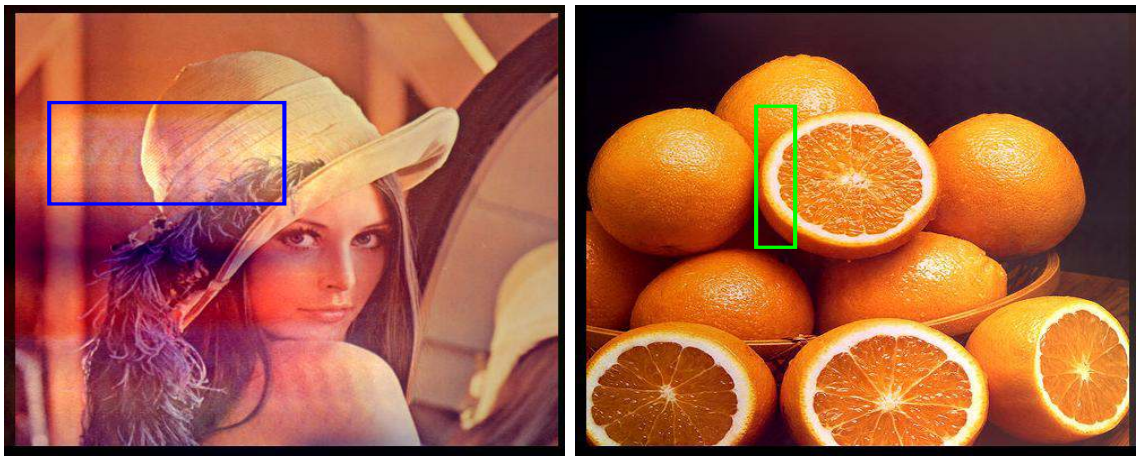


Figure 2.10: *Reconstruction of Lenna and Oranges generated from using the Weiss et al. [94] separation approach on the ten frame sequence in figure 2.9 (left column). Lenna is reconstructed by applying the Weiss approach on the stabilized sequence over Lenna (shown in figure 2.9, middle column). Oranges is reconstructed by applying the Weiss approach on the stabilized sequence over Oranges (shown in figure 2.9, third column). The white parts of the orange skin (for example see green rectangle, right) dominated the mixtures at their corresponding sites. This generated slight visual artifacts during Lenna's reconstruction (see blue rectangle, left). Pictures are scaled by a factor of 2 (left) and 1.5 (right) for illustration clarity.*

sequences with simple repetitive background motion. This motion was estimated by extracting the periodic global motion of the sequence.

Using Weiss' implementation[1], figure 2.10 shows the separation results generated from applying Weiss et al. approach on the ten frame sequence shown in figure 2.9 (left column). To reconstruct a specific layer, the sequence is first stabilized over the examined layer and then Weiss is applied to this stabilized sequence. Figure 2.9 (middle and right columns) shows the stabilized sequences over Lenna and Oranges respectively. Figure 2.11 shows the reconstructed static background of a 30 frame real sequence using the Weiss et al. separation approach. As shown the white reflections (shown in red, first row) are removed from the final background reconstruction (last row).

2.3 Multiple Motion Estimation

Most motion estimators assume the presence of one motion vector per pel [37,48]. However, for regions of reflections two motions exist, one for the foreground layer and one for the background layer. Motion estimators for regions of reflections contain two main stages. The first estimates the two motion vectors for each examined site in the observed mixture while the second assigns

[1] www.cs.huji.ac.il/ yweiss/

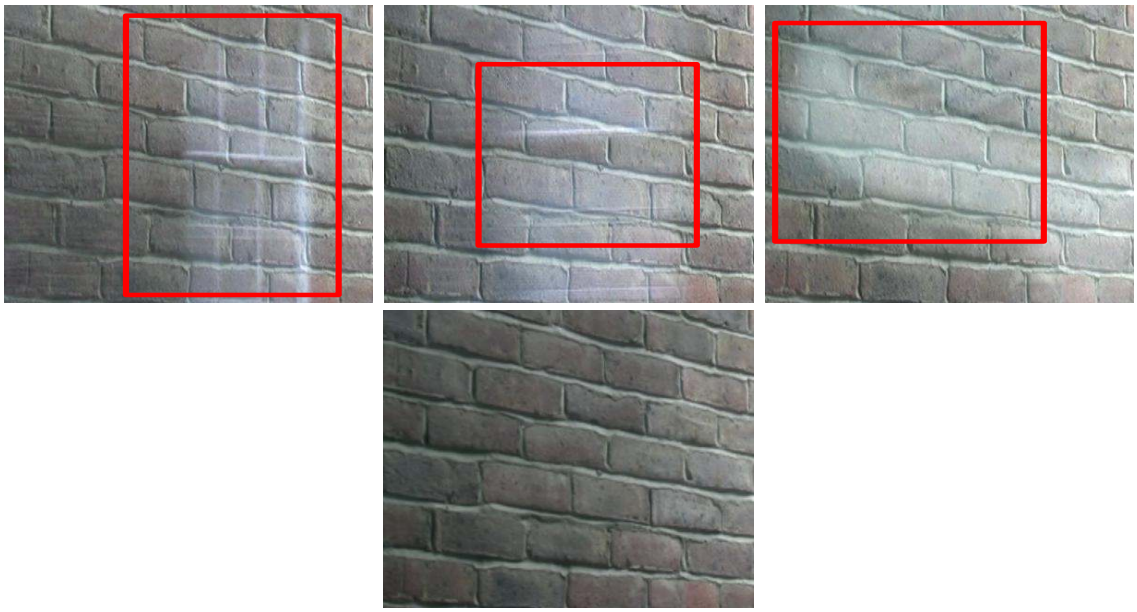


Figure 2.11: *Top, from left: Frames 1, 9, and 24 from a 30 frame sequence of a swinging glass door. In the sequence different objects are reflected on the swinging glass door (such as the ones shown in red) while there is a static background layer of brown bricks behind the door. The visual impact of this set up is the mixing between the background layer with the different foreground objects (shown in red) reflected on the glass door. Last row: Reconstructed background layer generated from applying Weiss et al. [94] on the 30 frame sequence.*

each vector to its layer. We call the second stage the 'motion-layer labeling problem'.

A large body of work exists on estimating the multiple motion vectors [3,7,59,61,75,81,87,88] while few authors extended their work to explicitly assign vectors to their corresponding layers [7, 81, 87]. Algorithms for estimating the vectors are divided into two main categories, Optic Flow and Fourier-Transform based. Both approaches treat reflections M as a linear combination between the foreground and background layers F and B as follows

$$M = F + B \tag{2.21}$$

Here the mixing parameters are assumed to be constant through the examined sequence and hence F and B denote the scaled versions of the original foreground and background layers. Using equation 2.21, Optic Flow approaches [3, 59, 75, 87] extend the brightness consistency assumption of Horn et al. [37], also known as the Optical Flow Constraint Equation (OFCE), to two motions. This approach however requires constant motion over three frames and can only handle small displacements.

Fourier-Transform based approaches are based on expressing equation 2.21 in the Fourier Domain. Using this approach Vernon [88] showed how to separate the two source layers and to calculate their corresponding motions assuming constant motion over five frames. This approach

was later extended by Stuke et. al [81] to estimate the two motion vectors of the source layers assuming constant motion over three frames. Exhaustive search using Block Matching is used to calculate the motion vectors. This approach however is not robust to temporal motion inconsistencies and is computationally expensive.

Toro et. al [87] proposed an approach for assigning the calculated motions to their corresponding layers by fitting two global motion models to the source layers. Later, Stuke et. al [81] proposed to solve this problem through a Bayesian framework. Motion vectors are modeled as MRFs and they are assigned to their layers in a way such that the local motion smoothness within each layer is maximized. To generate temporally consistent results, they impose temporal consistency on the motion vectors. Auvray et. al [7] extended this approach to handle more than one motion model per layer. Interesting results are shown in both [7,81] however this approach is not robust to temporal motion inconsistencies and can lead to error propagation over time.

In the remaining part of this section we discuss the Optic Flow and Fourier-based approaches for multiple motion estimation in more detail. We show results on processing real data and illustrate the problems with both approaches.

2.3.1 Optic Flow Approaches

Optic Flow approaches use the well known Optical Flow Constraint Equation (OFCE) originally proposed by Horn et. al [37]. For a single layer (say F) moving u pels/frame, OFCE states that

$$\frac{dF}{dt} = 0 \tag{2.22}$$

This equation can be rewritten as follows

$$\frac{dF}{dx}\frac{dx}{dt} + \frac{dF}{dy}\frac{dy}{dt} + \frac{dF}{dt} = u_x F_x + u_y F_y + F_t = 0 \tag{2.23}$$

where $[u_x, u_y]$ are the x and y motion components of u. This equation can be rewritten as $d(u)F(x,t) = 0$ where $d(u) = u_x d_x + u_y d_y + d_t$. Solving for u directly from equation 2.23 generates very small estimates of the motion components as equation 2.23 assumes that u is close to zero. Approaches for estimating the correct motion are mainly based on refining the estimates of u in an iterative manner [48]. At each iteration the layer F is shifted with the updated motion estimate. The updated motion is taken as the summation of the motion estimates up to the current iteration. OFCE is then applied on the currently shifted image and motion is recalculated and updated. This process is usually repeated several times until motion estimation error goes below some value.

In order to extend equation 2.23 to multiple motions, we assume the examined image is a mixture between two layers. For another layer (say B) moving with $[v_x, v_y]$, its own OFCE is $d(v)B(x,t) = 0$. As reflection M is expressed as an addition of two layers F and B, $d(u)$ and $d(v)$

can commute on F and B and hence the OFCE for two motions become $d(u)d(v)M(x,t) = 0$. This simplifies to

$$u_x v_x M_{xx} + u_y v_y M_{yy} + M_{tt} + (u_x v_y + v_x u_y) M_{xy} + \\ (u_x + v_x) M_{xt} + (u_y + v_y) M_{yt} = 0 \tag{2.24}$$

Here the term $M_{tt} = M_1 - M_2 - M_3$ and hence it implies that motion is assumed to be constant over three frames. Some approaches [3, 59, 87] were proposed to solve equation 2.24. They assume that motions are constant within an image patch and estimate $[u, v]$ by minimizing the left hand side of equation 2.24 within the examined patch. Aach et al. [3] estimates motions by eigensystem analysis of suitably extended tensors, yielding so-called mixed-orientation parameters (MOPs). They show how to decompose the MOP vectors into the individual motions. They show interesting results on mixtures formed by highly textured layers. Mota et al. [59] split equation 2.24 into linear and non-linear parts. Similarly to Aach et al. [3], the linear part is solved by eigenvalues analysis of a suitable structure tensor. A closed form solution is then proposed to solve the non-linear part by expressing motions as complex numbers. They show interesting motion estimation on one real sequence. Finally, Toro et al. [87] solve for the motions by minimizing the residual of equation 2.24 using the Geman-McClure norm. Optic Flow approaches show interesting results on a limited number of real sequences. However, they assume constant motion over three frames and hence cannot handle temporally active motions such as those arising due to acceleration or camera shake. In addition, equation 2.24 used here is just valid for small image displacements. Authors have solved this problem for single motion estimation by iteratively refining motion estimates and shifting layers with the updated motions [48]. Hence for reflections we need to shift each layer by its own motion estimate. This however is not an easy task as it requires estimating the examined foreground and background layers before we can shift them separately. No author has addressed this problem and the vast majority of the existing layer separation techniques cannot perform separation from just one image mixture as required here.

Figure 2.12 (left column) shows the estimated motions using the Optic Flow approach for a region containing a mixture between two different objects, each object moving with a different motion. The foreground and background objects are shown in green and blue respectively. Both the background and foreground objects are moving to the right however the background is moving with a faster speed. The yellow and red vectors in figure 2.12 show the estimated motions for the background and foreground objects respectively. Here we use the motion estimator of Mota et al. [59] which uses the Optic Flow approach. Mota et al. estimates two motions per site but does not assign each motion to its corresponding layer. Therefore we use Stuke et al. [81] Motion-Layer labeling approach to assign the estimated vectors to their corresponding layers. Using this technique motion estimates are shown in figure 2.12 (left column). As can be seen the estimated motions are significantly smaller than the groundtruth. This is expected as the Optic Flow approaches are based on equation 2.24 which requires the examined regions to be

undergoing very small motions.

2.3.2 Fourier Transform Based Approaches

Fourier Transform Based approaches are based on transforming equation 2.21 to the Fourier Domain. Assuming constant motions $[u, v]$ for the source layers F and B respectively, the examined mixture at time t and site $\mathbf{x}$ is expressed as follows

$$M(\mathbf{x}, t) = F(\mathbf{x} - tu) + B(\mathbf{x} - tv) \tag{2.25}$$

In the Fourier domain equation 2.25 becomes

$$\mathcal{M}(\omega, t) = (\phi_f)^t \mathcal{F}(\omega) + (\phi_b)^t \mathcal{B}(\omega) \tag{2.26}$$

where $\phi_f = \exp^{-j\omega u}$ is the phase shift for F at $t = 1$, ω is frequency, and $[\mathcal{F}, \mathcal{B}]$ are the Fourier transform of F and B.

Vernon [88] solves for the phase shifts and the source layers by substituting $t = [0 : 3]$ in equation 2.26. This generates a linear system of four equations and four unknowns being ϕ_f, ϕ_b, $\mathcal{F}(\omega)$, $\mathcal{B}(\omega)$. The Hough transform is then used to extract the motions $[u, v]$ from the estimated phase shifts. They show interesting motion estimation and layer separation results on real sequences, however, their approach requires the motion of each layer to be constant over five frames.

Stuke et al. [81] showed that it is possible to use the Fourier domain expression of the problem to derive a recursive expression for the dependency between more than one frame. For two motions the expression is

$$M_2(\mathbf{X}) + M_0(\mathbf{X} - u - v) - M_1(\mathbf{X} - u) - M_2(\mathbf{X} - v) = 0 \tag{2.27}$$

Stuke et al. [81] and Auvray et al. [7] then solve for (u, v) by minimizing the left hand side of equation 2.27 using exhaustive search through block matching. Here (u, v) are assumed to be spatially constant within a small image patch. They show good motion estimation results on synthetic and real sequences. However, one major drawback of this approach is that it assumes constant motion over three frames and hence leads to error propagation in the case of temporally active motion or temporal motion inconsistencies. In addition, it is computationally expensive as N^4 different combinations of (u, v) are examined if we search N possible displacements for each motion component $[u_x, u_y, v_x, v_y]$.

Figure 2.12 (right column) shows the estimated motions using the Fourier-Transform Based approach for a region containing a mixture between two different objects. Here motions are estimated using the Stuke et al. technique [81]. Stuke et al. estimates two motions per site and assign each motion to its corresponding layer. However it requires each object/layer to have a constant motion over a window of three frames. This however is not the case with the examined sequence as it is shot with a hand camera and hence it is a bit shaky. This generated

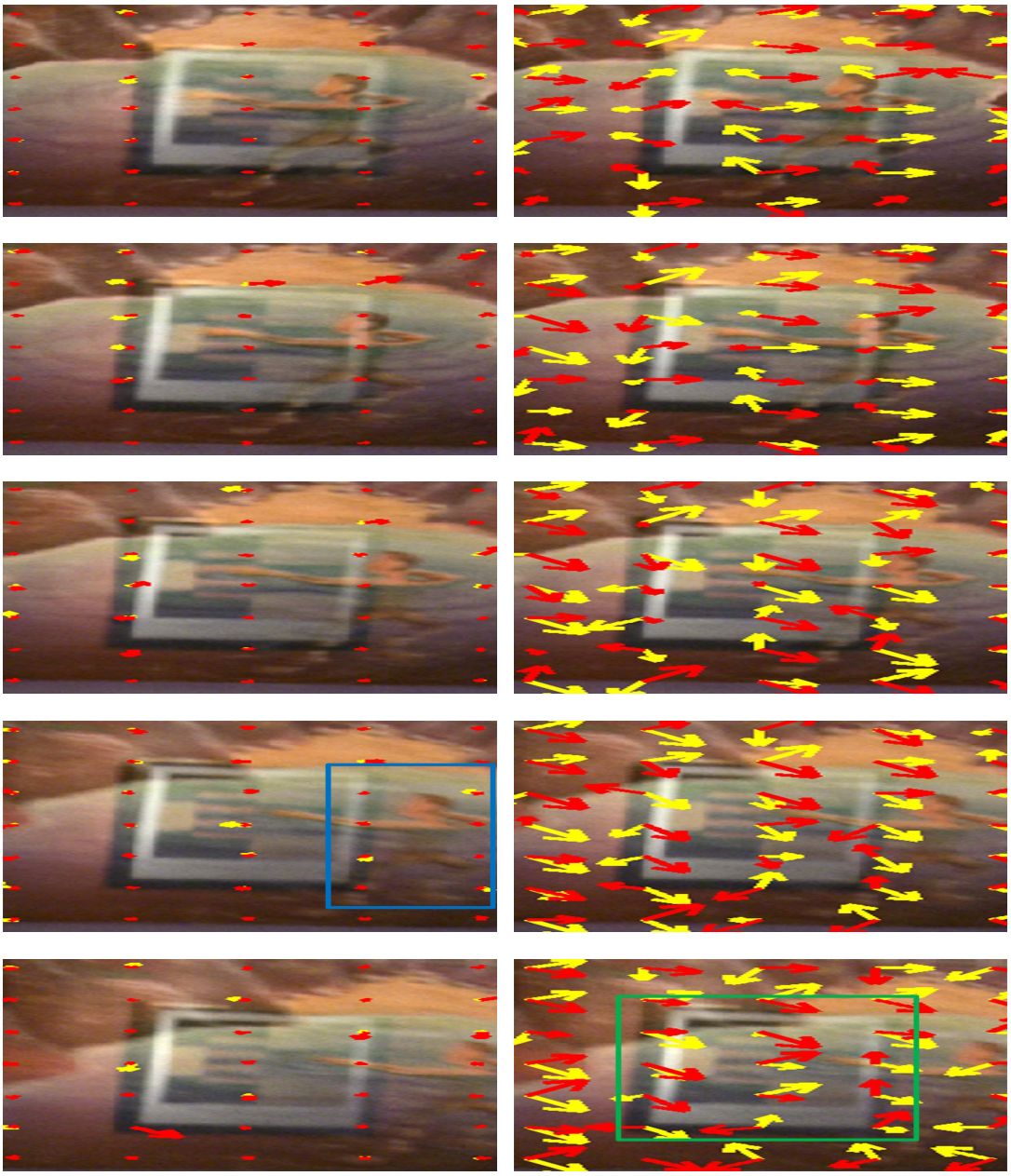


Figure 2.12: *From top; Each column shows 5 consecutive frames from an image sequence containing a reflection of a foreground object (shown in green) over a background object (show in blue). The foreground here is a rectangular shaped object while the background is a picture of a man-like drawing. The foreground is moving to the left while the background is moving to the right. The scene is shot with a hand camera and hence it is a bit shaky. The yellow and red vectors show the estimated motions for the background and foreground objects respectively as calculated by Mota et al. [59] (left column) and Stuke et al. [81] (right column). All motions are scaled by* 5 *for illustration clarity. It is clear that both approaches failed to estimate the correct motions of the foreground and background objects.*

erroneous motion estimates in many forms. First, the estimated motions hardly point to the correct directions of the motions. In addition, estimated motions for each layer are spatially and temporally inconsistent. The main reason for this spatio-temporal inconsistency is due to error propagation as the Stuke et al. approach imposes temporal consistency on the estimated motions. Hence, motion errors propagated through frames as they were incorrectly estimated at an earlier stage.

2.4 Conclusion

This chapter discussed previous work on handling transparency in natural images. Transparency is often the result of mixing a foreground layer with a background layer. 'Opacity' is the term used in the literature to describe the visibility degree of the foreground layer. Three main problems were addressed in this chapter: 1) Matte Extraction 2) Layer Separation and 3) Multiple Motion Estimation for Regions of reflections.

The term 'Matte' is given to the soft blend between the foreground and background objects that exist between object borders. All matting approaches require an initial segmentation of the examined image into three main regions being; 1) definite foreground 2) definite background and 3) unknown. Matte extraction techniques solve for the opacity values in the unknown region. We discussed Chuang et al. [17] Bayesian Matting in detail as it is used in our proposed restoration technique. Unlike most of the other matting techniques, Bayesian Matting uses a Bayesian Framework and hence it can be extended to other applications e.g. restoration.

For layer separation we discussed techniques designed for still images as well as image sequences. Most still image techniques require two mixtures of the same foreground and background layers. We discussed the Sarel et al. [73] 'Layer Information Exchange' optimization approach which estimates the optimal mixing parameters. This approach exchanges information between the reconstructed layers where optimal layer reconstruction is defined by the point where the structural similarity between the reconstructed layers is minimized. Two main approaches for measuring the structural similarity between the separated layers were discussed. The first is using GNGC of Sarel et al. [73] which measures the grayscale correlation between the reconstructed layers. The second approach is to assess the sparseness of edges or corners in the separated layers. Here the correct foreground-background candidate from a set of candidates is defined as the one with minimum corners or edges. We showed how layer separation by minimizing corners/edges sparseness could generate better separation than using GNGC. For image sequence separation techniques, we discussed the Weiss et al. [94] approach. This technique reconstructs a static (background) layer that is undergoing illumination variation over time. Illumination variations are due to the movement of different foreground layers over the background.

The last part of this chapter discussed previous work on estimating the motions of the foreground and background layers in regions of reflections. Two main approaches exist for

motion estimation 'Optic Flow' and 'Fourier-Transform Based'. Optic Flow approaches can only estimate very small motions. In addition, both approaches assume each layer is moving with a constant motion over at least three frames. Hence they cannot handle temporal inconsistencies arising due to small camera shake or motion acceleration. This often leads to erroneous motion estimates that propagate through time.

3

Blotch and Line Scratch Removal: A Review

Blotches and line scratches are two common degradations on archived footage. Automated removal of such corruptions is important in film restoration and typically involves a detection followed by an interpolation step. Most of the current algorithms model the corruption as a binary mixture between the original data and an opaque (dirt) field. Few algorithms however treat corruption as a semi-transparent (dirt) layer. When a semi-transparent model is used, interpolation becomes more robust to false alarms since clean data, if interpolated, remains less disturbed than if an opaque correction model is used.

Define the intensity of the pixel at site $\mathbf{x}$ in the nth frame of the observed corrupted sequence as $M_n(\mathbf{x})$, and in the clean original sequence as $B_n(\mathbf{x})$. Using the composting equation defined in earlier chapters the corrupted frame $M_n(\mathbf{x})$ at site $\mathbf{x}$ then becomes

$$M_n(\mathbf{x}) = \alpha_n(\mathbf{x})F_n(\mathbf{x}) + (1 - \alpha_n(\mathbf{x}))B_n(\mathbf{x}) \tag{3.1}$$

Here F is the corruption (dirt) layer and α is the corruption opacity with $\alpha = 1$ denoting complete obliteration of the original data. The traditional class of corruption removal techniques assume an opaque corruption layer, hence they only use binary values for the corruption opacity α_n. Thus the corrupted frame $M_n(\mathbf{x})$ at site $\mathbf{x}$ for binary corruption model becomes

$$M_n(\mathbf{x}) = \begin{cases} B_n(\mathbf{x}) & \text{for } \alpha_n(\mathbf{x}) = 0 \\ F_n(\mathbf{x}) & \text{for } \alpha_n(\mathbf{x}) = 1 \end{cases}$$

Some recent corruption removal techniques assume a semi-transparent corruption layer, they allow α to take non-binary values. For both corruption models, restoration starts by detecting

corruptions. Removal is then achieved by estimating the original data/background layer (B) for the detected sites. This is done using 'clean' information from the current and nearby frames.

This chapter starts by discussing two common approaches for blotch and line detection. However, as our thesis does not address the problem of corruption detection in an explicit form, we discuss those two detection approaches to show how false detections can be generated. This often leads to the generation of visual artifacts in clean regions when removal is performed using an opaque corruption model. We then go on to discuss previous work on blotch and line removal using opaque and semi-transparent corruption models.

3.1 Corruption Detection

3.1.1 Blotch Detection through Spike Detection Index

Kokaram [48] proposed the SDIp blotch detector. It detects blotches by looking for temporal discontinuities through image sequences. This is based on the observation that blotches are temporally impulsive events. Assume a clean pel $\mathbf{x}$ in the examined corrupted frame $M_n(\mathbf{x})$ undergoes a displacement of $\mathbf{d}_{n,n-1}$ from the previous frame. One can then relate $M_n(\mathbf{x})$ with M_{n-1} as follows

$$M_n(\mathbf{x}) = M_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}) + e(\mathbf{x}) \tag{3.2}$$

Here $e(\mathbf{x})$ denotes the error in this simple motion model which is also known as the Displaced Pixel Difference (DPD). However as blotches are temporally impulsive events, they violate this motion model and generate large DPD values. In addition, as blotches do not propagate through frames, they generate high DPDs for both forward and backward directions. SDIp blotch detector of Kokaram et al. [48] is based on these observations and is stated as follows

$$\alpha(\mathbf{x}) = \begin{cases} 1 & \text{When } (|E_b| > E_t) \text{ AND } (|E_f| > E_t) \text{ AND sign}(E_f)\text{=sign}(E_b) \\ 0 & \text{otherwise} \end{cases}$$

Here E_b and E_f are the backward and forward DPD's and E_t is a user-defined detection threshold. The last term of SDIp is based on the observation that blotches generate forward and backward DPD's of the same sign. This is based on the work of Storey [80] where he correctly noted that blotches often occupy intensities well outside the intensity of the surrounding region.

Equation 3.2 assumes simple motion between frames and hence it is violated in regions of complicated motion. This often generates high false detection in such regions. Figure 3.1 shows the result of running SDIp on two frames. Here E_t is set to 5 for both frames. In the first frame (see figure 3.1, top row) there is little motion between frames. This helped in generating good blotch detection with low false detection rate. However in the second frame (see figure 3.1, bottom row) the actors are moving quickly. This generates large Displaced Frame Difference (DFD) in uncorrupted regions. The result is high false detection rate. In both frames, detection of true blotches goes beyond the corruption borders. This flags clean regions around blotches as

Figure 3.1: *Left, from top: Frames 58 and 78 for a corrupted image sequences with blotches shown in green. Right: Corresponding SDIp detections. Here E_t is fixed to* 5*. For frame 78 the actors are moving quickly. This generates high false blotch detection rate.*

corrupted. Such examples of false detections will lead to the correction of uncorrupted regions during the removal stage. This will generate unnecessary restoration artifacts in uncorrupted regions.

3.1.2 Line Detection

Since this thesis is concerned with corruption removal rather than detection, we do not present in this section a literature review of line detectors. However what we show here is that line detection can have false alarms. We show that by discussing a commonly known line detector developed by Kokaram [48]. False line detections will effect the removal step.

Kokaram Line Detector: The earliest work in line detection is by Kokaram [48]. It is based on the observation that line scratches are near-vertical lines causing horizontal spatial discontinuities (see figure 3.2, first row). A horizontal spatial derivative is first applied to the

examined frame followed by simple thresholding. This generates a binary mask for potential line scratches. We call the generated binary mask BM. Kokaram detects line scratches by looking for near vertical lines in BM. This is done by using the Hough transform. This approach however can misclassify true vertical lines.

Figure 3.2 shows examples on detecting line scratches using this approach. BM is generated by applying a threshold of 2 on the filtered frame. In the left frame the line scratch is correctly detected (shown in green, left column). It corresponds to the highest peak in the corresponding $[\theta, r]$ space (last row, left, shown in green). However, in the other frame (right column) line scratches exist (shown in green) as well as some true vertical lines (shown in red, first row). Here the correct line scratch (shown in green, first row) represent the 4^{th} strongest peak in the $[\theta, r]$ space (see last row, right, shown in green). The three highest peaks however correspond to three true vertical lines (shown in red, first row). This incorrectly classifies the true vertical lines as scratches. In both frames it is hard to locate the borders of the line scratches. This often makes detection of true scratches goes beyond the corruption borders. Such examples of false detections will lead to the correction of uncorrupted regions during the removal stage. This will generate unnecessary restoration artifacts in uncorrupted regions.

3.2 Restoration Using Opaque Corruption Model

Previous approaches for removing blotches and line scratches using an opaque corruption model have largely fallen into three main categories: Image Inpainting based, Heuristics based and Model based. All approaches estimate the original data in corrupted regions using clean information from the examined and sometimes nearby frames.

Image Inpainting based approaches treat corruptions as missing holes in images [10, 21, 24, 25, 28, 43]. Those holes are filled using nearby spatio-temporal 'clean' data. Many inpainting techniques are based on Efros et. al's work for texture synthesis [25] and Bertalmio et al.'s work on inpainting using total variation minimization [10]. However, as most of these techniques do not address corruption removal in an explicit form, they do not incorporate any information specific to the type of the examined corruption. Heuristic based approaches remove corruptions by using median filters as they are robust to impulsive noise [4–6, 34, 51, 57, 80]. Model based approaches explicitly model corruptions as an opaque layer superimposed on the original layer [47–51]. In addition they use spatial and temporal priors specific to the type of the examined corruption and the underlying original data.

This section starts by describing Efros et al.'s [25] Texture Synthesis technique in the context of blotch removal. It then discusses state of the art blotch removal techniques using Heuristics and explicit corruption modeling. Last, it outlines modifications for line removal.

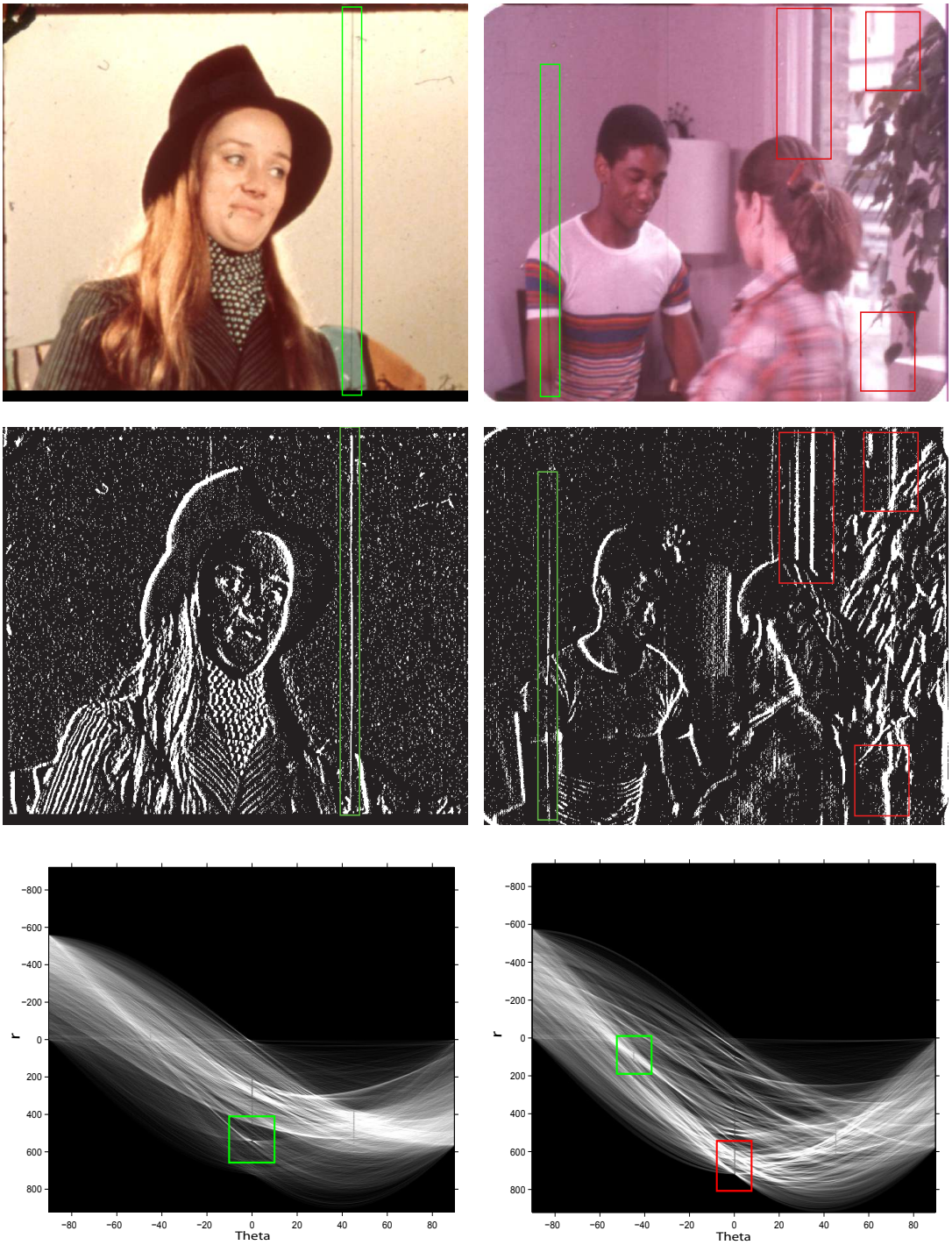


Figure 3.2: *From top: Frames corrupted with lines scratches (shown in green); the corresponding spatial horizontal gradient mask BM. This mask is obtained by flagging a spatial horizontal derivate of more than 2 grayscale difference as possible corruption;* $[\theta, r]$ *space generated by applying the line Hough transform for regions where* $BM = 1$*. In the first two rows green and red represent correct and incorrect scratch detections respectively. In the last row the green and red boxes correspond to the* $[\theta, r]$ *peaks of correct and false line detections respectively.*

3.2.1 Texture Synthesis for Blotch Removal

The technique of Efors et al. [25] can be used to remove blotches by filling the missing data one pel at a time. The corrupted region is filled in an order such that the isophote lines arriving at the blotch borders are connected inside the corrupted region. This filling order was proposed by [10] and is shown to reduce reconstruction artifacts. To estimate the original value for a corrupted pel, block matching is performed between the examined pel and the nearby 'clean' data. A block is centered on the examined pel and is matched with nearby blocks by minimizing the SSD. Here block matching is only performed on the uncorrupted part of the blocks. The center of the block generating the best match is then taken as the estimated value for the examined pel. This process is performed over all sites detected as blotches. Bornard et al. [11] was the first to apply similar texture synthesis ideas to restore archived footage. They estimated the original data using clean data from nearby frames as well as clean data from the current frame. They showed interesting results on removing blotches.

One major disadvantage with corruption removal using texture synthesis is the high computational complexity. Here block matching is performed for each pel detected as corrupted. This could lead to explosion in computational time if the generated blotch detection masks give a high false detection rate. In addition, the technique treats the whole blotch detection mask as corrupted. As a result uncorrupted regions misclassified as corrupted will be processed. This could generate unnecessary removal artifacts in uncorrupted regions.

Figure 3.3 shows examples of removing blotches using our implementation of Efros's texture synthesis technique. In our implementation we estimate the original data by examining the clean data in the current, next and previous frames. Temporal information generates more candidates for the original data and hence improves reconstruction. Figure 3.3 (left column) shows that Efros can remove blotches successfully. However, figure 3.3 (right column) shows an example where Efros generates restoration artifacts in uncorrupted regions misclassified as corrupted by the blotch detector (shown in green).

3.2.2 Heuristics Based Approaches

Heuristic based approaches are based on the fact that blotches are impulsive events in space and time. Storey [80] was the first to use this observation for blotch removal by using a 3-tap non-motion compensated median filter. Alp and Arce [4–6] introduced a 3-D multilevel non-motion compensated median filters (ML3D) for removing blotches. Alp et al. estimates the value of an examined pel as the median of the output of three median filters. The filter masks for the three median filters are shown in figure 3.4 (top). This technique was later modified by Kokaram et al. [51] into ML3Dex by adding two more median filters (see figure 3.4, bottom).

Figure 3.5 shows the result of processing a frame with ML3Dex. The actors are moving fast generating high false detection rates. Blotches are correctly removed (shown in green) however restoration artifacts are generated due to processing uncorrupted regions detected as corrupted

Figure 3.3: *Removing blotches (shown in black boxes) from frames 62 (left column) and 83 (right column) from an image sequence using our implementation of Efros et al. [25]. We use temporal as well spatial information in the reconstruction process. From top: Corrupted frames; SDIp masks generated with a threshold of 8 grayscale differences; Generated reconstruction. Blotches in frame 62 (left) are removed successfully with hardly any noticeable reconstruction errors. However, there are visible reconstruction artifacts in frame 83 (shown in green). Here the color stripes on the actor's shirt are nearly erased (compare the green box in the bottom row with blue box at top row). This is due to processing uncorrupted regions misclassified as corrupted by SDIp.*

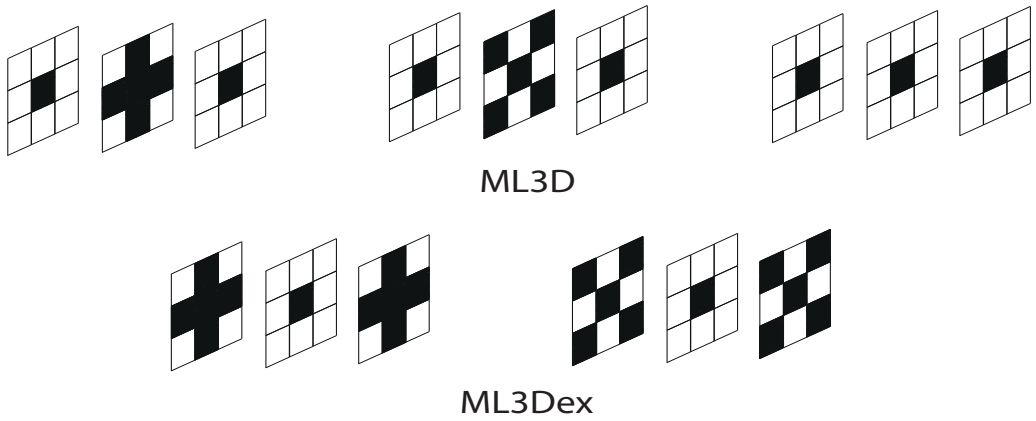


Figure 3.4: *Top: Filter masks for ML3D. Each column represent one of the three ML3D median filters and the masks for a filter are applied for the previous, examined and next frames respectively. Bottom: The two extra filter masks of ML3Dex.*

by SDIp (shown in blue).

3.2.3 Model Based Approaches

Model based approaches for blotch removal traditionally model blotches as an opaque corruption layer superimposed on the original layer as follows

$$M_n(\mathbf{x}) = \alpha_n(\mathbf{x})F_n(\mathbf{x}) + (1 - \alpha_n(\mathbf{x}))B_n(\mathbf{x}) + \mu(\mathbf{x}) \tag{3.3}$$

Here $M_n(\mathbf{x})$ is the observed value for pel $\mathbf{x}$ at frame n, $[B, F, \alpha]$ are the corresponding original data, blotch values and blotch opacity respectively and $\mu \sim \mathcal{N}(0, \sigma_\mu^2)$ is the observation model noise. Here α is only allowed binary values. Kokaram et al. used this model to propose a variety of Bayesian techniques for the joint solution of motion and degradation removal [47–50]. Their techniques solve for motion as they noticed that performing a motion-compensation stage for blotch removal improves removal quality significantly. This however comes with high computational cost.

Motion information is estimated using the single motion model previously used for the SDIp detection which is re-stated here as follows

$$M_n(\mathbf{x}) = M_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}) + e(\mathbf{x}) \tag{3.4}$$

Here $\mathbf{d}_{n,n-1}$ is the motion between the examined frame and the previous frame for site $\mathbf{x}$ while $e \sim \mathcal{N}(0, \sigma_\mu^2)$ is the observation model noise. This model however fails when the observed motion is pathological, e.g. very fast, containing motion blur, or periodic elements. In that case the reconstruction process introduces more defects than it removes. This is a problem particularly with the movement of people and clothing. Roosmalen, Rares, Bornard, Kent, Corrigan et

Figure 3.5: *Removing blotches from frame 67 of an image sequence using ML3Dex [51]. Top; Original frame (left) with blotches shown in green and the corresponding SDIp mask (right). Here a SDIp threshold of 8 grayscale differences is used. Bottom: ML3Dex reconstruction. Blotches (shown in green, top) are removed however restoration artifacts are generated in uncorrupted areas (shown in blue).*

al. [11, 19, 44, 63, 68] introduced several mechanisms for dealing with such pathological motion effects. Roosmalen concentrated on detecting failure in the motion estimator he used (based on Phase Correlation) by simply turning off the blotch remover when the DFD was too high over many consecutive frames. Kent et al [44] simply turned off the blotch remover in any moving foreground region detected by a crude image object segmentation process based on motion. The idea here was that motion estimation typically fails in foreground regions (if it fails at all) and so this yielded a very conservative process. Bornard and Corrigan pushed Roosmalens ideas into a Bayesian framework, eventually incorporating MRF priors for blotch smoothness in time that allowed the blotch remover to implicitly disable itself when a blotch was being detected in more than one frame consecutively. Rares used machine learning type ideas to detect picture material

which was difficult for motion estimation, again turning off the process in difficult areas.

JONDI [47] is Kokaram's latest technique for blotch removal. It uses most of the ideas proposed in his earlier work and improves on them [48–50]. JONDI's main aim is to address the interactions between the three main stages of blotch removal. Those stages are Motion Compensation, Detection, and Interpolation. Poor motion estimates generate poor (say SDIp) detections. Poor detection generally leads to poor reconstruction since the detector is probably flagging areas which cannot be properly reconstructed. JONDI is designed to reduce the effect of motion estimation failure for blotch removal. Many motion errors arise due to object occlusion and uncovering. JONDI addresses motion estimation failure for blotch removal by explicitly incorporating models that allow for occlusion and uncovering. In addition it models $[\alpha, B]$ as MRFs and imposes smoothness on the generated reconstructions.

3.2.3.1 Joint Noise, Detection and Interpolation: JONDI

To account for occlusion in motion estimation, a hidden field of binary variables is introduced between the examined frame n and the previous and next frames $n-1$ and $n+1$ respectively. Each pel in the examined frame is assigned to a state in the form of $[\alpha(\mathbf{x})\ \ O_b(\mathbf{x})\ \ O_f(\mathbf{x})]$ as follows

000 The pel is not corrupted and there is no occlusion
001 The pel is not corrupted and there is forward occlusion
010 The pel is not corrupted and there is backward occlusion
100 The pel is corrupted and there is no occlusion
101 The pel is corrupted and there is forward occlusion
110 The pel is corrupted and there is backward occlusion

The system then derives an estimate for $[B, \alpha, F, O_f, O_b, \mathbf{d}_{n,n-1}, \mathbf{d}_{n,n+1}]$ for each pel $\mathbf{x}$ from the posterior $P(B_n, \alpha, F, O_f, O_b, \mathbf{d}_{n,n-1}, \mathbf{d}_{n,n+1} | M_n, B_{n-1}, B_{n+1})$. The posterior is factorized in a Bayesian fashion as follows (where $\mathbf{x}$ is dropped for clarity)

$$\begin{aligned} P(B_n, \alpha, F, O_f, O_b, \mathbf{d}_{n,n-1}, \mathbf{d}_{n,n+1} | M_n, B_{n-1}, B_{n+1}) \propto & P(M_n | B_n, F_n, \alpha) \\ & P(B_n(\mathbf{x}) | B_{n-1}(\mathbf{x}), B_{n+1}(\mathbf{x})) P(B_n | B_n^{\mathcal{N}}) \\ & P(\alpha_n | \alpha_n^{\mathcal{N}}) P(F_n | F_n^{\mathcal{N}}) P(O_f | O_f^{\mathcal{N}}) P(O_b | O_b^{\mathcal{N}}) \\ & P(\mathbf{d}_{n,n-1} | \mathbf{d}_{n,n-1}^{\mathcal{N}}) P(\mathbf{d}_{n,n+1} | \mathbf{d}_{n,n+1}^{\mathcal{N}}) \end{aligned} \tag{3.5}$$

The first term measures the error in the observation model of equation 3.3, the second term measures the error in calculating the original data B, while the remaining terms impose spatial smoothness on the estimated parameters for $[B, \alpha, F, O_f, O_b, d_{n,n-1}, d_{n,n+1}]$. $\mathcal{N}$ here is a local spatial neighborhood from $\mathbf{x}$ on which smoothness is imposed.

Solving for the Missing Parameters: Motion estimates are initialized with simple block matching while α are initialized with SDIp. The examined frame is scanned in the pel basis

using the current motion and α estimates, generating an estimate for $[B, O_f, O_b]$. A simple way of configuring the occlusion fields is by flagging sites with high backward/forward DFD as ones undergoing occlusion in the previous/next frames. A more complicated approach of configuring the occlusion fields is by integrating out B and α from the posterior (equation 3.5) and maximizing the resulting function with respect to all possible combinations of the occlusion fields (see [47] for more details). Proceeding with solving for the missing parameters, given the current estimates for $[B, O_f, O_b]$, motion estimates and α are refined. This process is iterated until reasonable pictures are built. Here the quality of the reconstructed pictures is assessed by evaluating the posterior of equation 3.5 at each iteration (see [47] for more detail).

Figure 3.6 shows the result of processing three consecutive corrupted frames with JONDI. Here the actors are moving their hands very fast causing occlusion and uncovering of the guy's hand/shirt. JONDI removed most of the corruption successfully (shown in green). However, it generated restoration artifacts in uncorrupted regions undergoing fast motion (shown in blue). The main reason for such failure is that JONDI treats the whole corruption area as opaque and hence processes clean data that is incorrectly detected as corrupted.

3.2.4 Modifications for Line Removal

Unlike blotches, line scratches have smaller width (in range of 3-7 pels) and they exist at the same location in several frames. Due to their smaller width, blotch removal approaches that use spatial information can be applied directly for line removal. As such techniques estimate the original data using clean data close to the corrupted sites, they generate better removal when corruptions span small areas as more clean information will be available. Line removal techniques using image inpainting often fill the corrupted region in a way such that the isophote lines arriving at the corruption borders are connected inside the corrupted region [28, 84]. This filling order was shown to improve image inpainting in general [10, 21] and is a good application to scratch removal due to their small width. Modifications for line removal then go on to extend image inpainting techniques to use clean information from nearby frames [28, 84]. To estimate the original value for a corrupted pel in frame n, a block is centered on that pel and is matched with blocks in the previous and next frames by minimizing the SSD. Here block matching is only performed on the uncorrupted part of the blocks. The center of the block generating the best match is then taken as the estimated value for the examined pel.

Kokaram [46] proposed a line removal technique (JOMBEI) that estimates the original data using spatial information from the examined frame. The algorithm generates a solution within a Bayesian framework similar to the one used in JONDI [47]. The main difference between JONDI and JOMBEI however is that JOMBEI mainly solves for the original data without using any motion information. It does so by modeling the underlying original data as a 2D-AR process. In the clean frame B the value of a pel $\mathbf{x}$ is modeled as a linear combination of nearby pels in

Figure 3.6: *Processing three consecutive corrupted frames (left) with JONDI. Reconstructions are shown on the right. Blotches are shown in green on the left. As shown on the right all blotches are successfully removed. However, restoration artifacts are introduced in uncorrupted regions undergoing fast motion (shown in blue).*

the same frame, as follows

$$B(\mathbf{x}) = \sum_{k=1}^{P} a_k(\mathbf{x})\mathbf{B}(\mathbf{x} + \mathbf{q_k}) + \mathbf{e}(\mathbf{x}) \tag{3.6}$$

Here a_k are weights and $e \sim \mathcal{N}(0, \sigma_e^2)$ is the observation model noise. The likelihood then becomes

$$P(M_n|B_n, F_n, \alpha) = \exp - \left(\frac{e(\mathbf{x})^2}{2\sigma_e^2}\right) \tag{3.7}$$

Spatial smoothness is imposed on the generated values for B through the 2D AR Model. The solution is generated as a sample from the posterior using Gibbs Sampling [46].

Figure 3.7 shows reconstruction results using three different line removal techniques all using spatial information only. Average filtering can generate blurred results as shown in figure 3.7 (a). Inpainting (Efros) can generate reconstruction artifacts in uncorrupted sites that are close to the examined scratch as shown in figure 3.7 (b) (in blue). The 2D-AR image model used in JOMBEI can generate blurred results as shown in figure 3.7 (c) (in blue). Here JOMBEI generates restoration artifacts by blurring the left part of the letter M.

3.3 Corruption Removal using Semi-transparent Corruption

Hisho et al. [38] in 1999 were the first to introduce a non-binary index for measuring the level of corruption for blotch removal in image sequences. Unlike opaque corruptions where a site is assigned a binary corruption value (1 for corrupted, 0 for clean), Hisho et. al proposed the assignment of a non-binary value instead. This value is a function of the temporal discontinuity between the examined site and the nearby frames. The function is learned through a training process [72]. Having calculated the corruption level for each site, the interpolated value is then set to an intermediate value between the observed image brightness and the output of an arbitrary spatio-temporal filter, depending on the corruption level. This is a simple process attempting to capture the true nature of the problem which should arise from consideration of equation 3.3 with non-binary opacity values as the degradation model.

Since the work of Hisho et al. [38] more researchers attempted to remove corruptions by modeling them as semi-transparent layers [13,14,20,29,32,78,79]. Greenblatt et al. [32] assumed that the original data hidden by the blotch was of a uniform color. They estimated that color in a way to match the color of the clean surrounding regions. Hence the original data B is estimated using the following equation

$$B(\mathbf{X}) = \left(\text{Med}[M(\mathbf{X})].\left(\frac{M(\mathbf{X})}{2 * \text{Med}[M(\mathbf{X})]}\right)^{\lambda}\right)^{\gamma} \tag{3.8}$$

where $B(\mathbf{X})$ is the estimated original data for region $\mathbf{X}$, λ and γ are fixed parameters and Med[.] is the median operator. As a final stage edges are corrected using a localized smoothing function which the author did not disclose in detail.

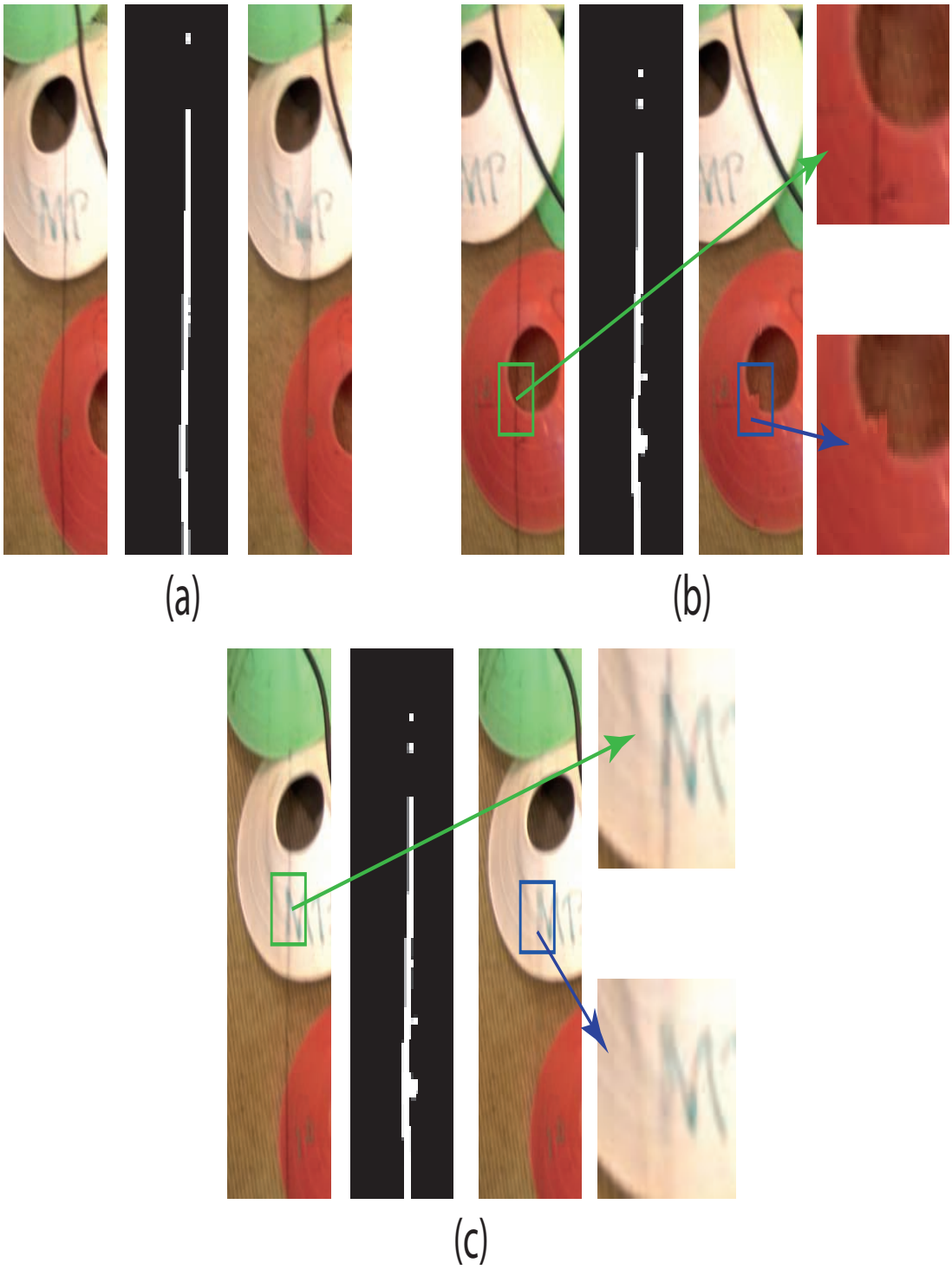


Figure 3.7: *Removing line scratches in three frames with (in clockwise direction); Average spatial filtering, Efros et al. [24] (our implementation) and JOMBEI respectively. For each technique we show (from left); corrupted data, the scratch mask, the reconstructed data respectively. Average filtering can blur the corruption, Inpainting can generate reconstruction errors such as the one shown in blue in (b) and JOMBEI can blur the underlying original data (see blue in c).*

Some work attempted to remove corruptions by explicitly modeling them as a linear mixture of original data and some corruption layer using equation 3.3 [13,14,20,29,78,79]. The corruption layer F is set to 0 in most techniques as they mainly address the removal of dark corruptions. Hence equation 3.3 for the examined frame becomes

$$M(\mathbf{x}) = (1 - \alpha(\mathbf{x}))B(\mathbf{x}) + \mu(\mathbf{x}) \tag{3.9}$$

Bruni et al. [13,14] used this model to remove line scratches by reducing the defect intensity till it is no longer visible. Here they use Weber's law to calculate the attenuation factor. Stanco et al. [78, 79] assumed a constant corruption opacity within a block of pels and estimated the original data within this block using the mean and variance of nearby clean data. Gai et al. [29] estimated the original data using clean data in the examined and nearby frames. Here they introduced a data term to capture the semi-transparent nature of the corruption. This term however does not relate the estimated opacity with the corruption model of equation 3.9. Instead, it just biases the estimated corruption opacities to be far from $\alpha = 1$ i.e. far from being opaque.

Crawford et al. [20] was the first to propose the use of equation 3.9 to explicitly calculate the corruption opacity and the original data for each pel detected as corrupted. Their work focuses on the removal of blotches on photographs due to moisture. They address the problem in the HSV color space and generate a solution within a Bayesian framework.

3.3.1 Crawford et al. Semi-Transparent Corruption Removal

Crawford et al. [20] first restore chroma components using a simple texture synthesis technique [25]. The luminance channel is then split into an over-complete wavelet representation. Wavelet details are left unchanged in the case of perfect semi-transparency or attenuated whenever dirt causes spurious edges. The approximation band is modeled as a linear mixture between dirt and the original data using the matting equation as follows

$$M_l(\mathbf{x}) = (1 - \alpha)B_l(\mathbf{x}) + e(\mathbf{x}) \tag{3.10}$$

Here $M_l(\mathbf{x})$ is the observed luminance approximation band at point $\mathbf{x}$, $\alpha(\mathbf{x})$ is the corruption opacity, $e \sim \mathcal{N}(0, \sigma_e^2)$ is the observation model noise and B_l is the luminance approximation band of the original data. Here the foreground corruption data F is set to 0 as the blotch in examination is assumed to be fully dark. Crawford et al. solves for $[B_l(\mathbf{x}), \alpha(\mathbf{x})]$ for every pel using Bayesian Matting and generates the final reconstruction by combining the reconstructed chroma and luminance components.

Bayesian approach: The system derives an estimate for $[B_l(\mathbf{x}), \alpha(\mathbf{x})]$ from the posterior $P(B_l, \alpha|M_l)$ (where $\mathbf{x}$ is dropped for clarity). This is factorized in a Bayesian fashion as follows

$$P(B_l, \alpha|M_l) \propto p(M_l|B_l, \alpha)p(\alpha|\alpha^{\mathcal{N}})p(B_l|B_l^{\mathcal{N}}) \tag{3.11}$$

The first term ensures that the estimated $[\alpha, B_l]$ resemble the observed value of M_l while the remaining terms impose spatial smoothness on the generated α and B_l. This is done by mod-

eling pels as 8-connected MRFs in the usual way and imposing smoothness within the local neighborhood $\mathcal{N}$. The terms of equation 3.11 are expressed as follows

$$\begin{aligned} p(M_l|B_l,\alpha) &\propto \exp - \left(\frac{(M_l(\mathbf{x}) - (1-\alpha)B_l(\mathbf{x}))^2}{2\sigma_e^2} \right) \\ P(B_l|B_l^{\mathcal{N}}) &\propto \exp - \left(\sum\nolimits_{k\in\mathcal{N}} |B_l - B_{lk}|^2 \right) \\ P(\alpha|\alpha^{\mathcal{N}}) &\propto \exp - \left(\sum\nolimits_{k\in\mathcal{N}} |\alpha - \alpha_k|^2 \right) \end{aligned} \tag{3.12}$$

Solving for $[\alpha, B_l]$: The algorithm starts by initializing B_l with values from nearby clean data and initializing α with the SDIp detection. $[\alpha, B_l]$ are then refined in an iterative manner using ICM until convergence.

3.4 Scope for a new removal technique

Recent work has introduced semi-transparency into blotch corruption models but none of the current semi-transparent removal techniques incorporate useful temporal information to estimate corruption opacity on pel basis. This often generates inaccurate estimation of the corruption borders which leads to the correction of uncorrupted sites. In addition most of the semi-transparency work has been directed at stills. Furthermore, current semitransparent removal techniques are either very sensitive to noisy measurements (Gai's [29]), cannot remove highly opaque corruptions (Saito and Crawford techniques [20,72]), can only handle very simple texture (Stanco, Greenblatt and Crawford [20, 32, 78, 79]) or require a detailed model of the corruption profile (Bruni techniques [13, 14]). There is no corruption removal technique adopting a semi-transparent model which can handle restoration in image sequences and can handle regions having complicated texture and undergoing fast motion. There is therefore a need for an algorithm that acknowledges the true state of media i.e. an image sequence corrupted by semi-transparent degradation.

4

Bayesian Inference for Semi-transparent Blotch and Line Removal

This chapter presents a new approach for removing blotches and line scratches from image sequences. Current removal techniques assume complete obliteration of the original data at the corrupted sites. This often leads to the introduction of restoration artifacts during removal. Our new technique is based on modeling corruption as a semitransparent layer. We define the opacity of a corruption in frame n at site $\mathbf{x}$ as $\alpha_n(\mathbf{x})$ and propose a new linear model of corruption as follows

$$M_n(\mathbf{x}) = \alpha_n(\mathbf{x})F(\mathbf{x}) + (1 - \alpha_n(\mathbf{x}))B_n(\mathbf{x}) \tag{4.1}$$

Here $M_n(\mathbf{x})$ is the observed corrupted intensity at a pixel site $\mathbf{x}$ in frame n, B_n is the clean original intensity at that site and F is a constant intensity representing the underlying corruption color. In our work we will use $F = 0$ to model black or rather dark blotches/lines. Hence the model implies that the observed data is a linear mixture between the clean original data and a corruption color F. Unlike most current removal techniques, in our technique α takes non-binary values where $\alpha = 0$ in clean regions and $\alpha = 1$ in sites where original data is completely obliterated.

Equation 4.1 is clearly related to the image layer model used in the vast quantity of Matting work (see Section. 2.1 for more details). In matte extraction an image is assumed to arise as the result of a linear mixture of foreground F and background B elements. The image is segmented into three regions being definite foreground, definite background and unknown. The matte extraction techniques solve for α for every point in the unknown region. Chuang et al.'s

Bayesian Matting (see Section. 2.1.1 for more details) solves for the F and B together with α. In our corruption model of equation 4.1, the corruption layer is the foreground, the original layer is the background, while the unknown region is the area detected as being corrupted. We use SDIp with a threshold of 5 to detect blotches (see Section. 3.1.1 for more details) and we detect line scratches manually as the examined lines are hard to detect using current line detection techniques.

Corruption removal is achieved by estimating the original (background) data B at corrupted sites. In our model F is known while $[\alpha, B]$ is unknown. We use Bayesian Matting to estimate $[\alpha, B]$ for every pel detected as corrupted. We use spatial and temporal priors in a way which maintains restoration integrity despite texture and motion complexity. We use a Bayesian framework which is related to the one used in Crawford et al.'s [20] blotch remover. Note that even though we do not address the problem of detection in an explicit form, the process of defect matte extraction proposed by our technique can be thought of as a detection refinement step.

In the next section we propose our Bayesian framework for semi-transparent corruption removal. We focus more on blotch removal. We then present modifications for line removal.

4.1 The Degradation Model

A corrupted pel $M(\mathbf{x})$ is modeled as a linear combination between the original data (background layer) $B(\mathbf{x})$ and the corruption field $F(\mathbf{x})$ (foreground layer) according to a blending factor α. This matting model was discussed previously and repeated here for clarity.

$$M(\mathbf{x}) = \alpha(\mathbf{x})F(\mathbf{x}) + (1 - \alpha(\mathbf{x}))B(\mathbf{x}) \tag{4.2}$$

Here $\alpha(\mathbf{x})$ is the mixing parameter where $\alpha = 1$ represents complete obliteration of the underlying data. In this work, dirt is assumed to be the source of the corruption and therefore F is fixed to an RGB value of $[0, 0, 0]$. This model is illustrated in figure 4.1 (top row) with a synthetic example. Note that we assume detection $d(\cdot)$ of the missing patches has already taken place, or at least some kick start is available. We use SDIp here. Hence small patches of $d(\mathbf{x}) = 1$ have been delineated and the problem then is to estimate the values of the original data B in these patches. There are therefore two unknown parameters at each corrupted pel being $[\alpha,B]$, and in a sense estimation of α amounts to a refinement of the kick start detection field $d(\mathbf{x})$.

4.2 Bayesian Framework

We estimate $[\alpha, B]$ for each corrupted pel from the posterior $P(\alpha, B|F, M_0, ...M_K, \alpha^{\mathcal{N}}, B^{\mathcal{N}})$ (where $\mathbf{x}$ is dropped for clarity). Here $[B^{\mathcal{N}}, \alpha^{\mathcal{N}}]$ are the unknown parameters within a local neighborhood $\mathcal{N}$ from the examined site while M_n is the n^{th} frame of the examined sequence which contains K frames. The posterior is factorized in a Bayesian fashion as follows

$$P(\alpha, B|F, M_0, ...M_K, \alpha^{\mathcal{N}}, B^{\mathcal{N}}) \propto P(M_u|\alpha, B, F)P(\alpha, B|M_0, ...M_K, \alpha^{\mathcal{N}}, B^{\mathcal{N}}, F) \tag{4.3}$$

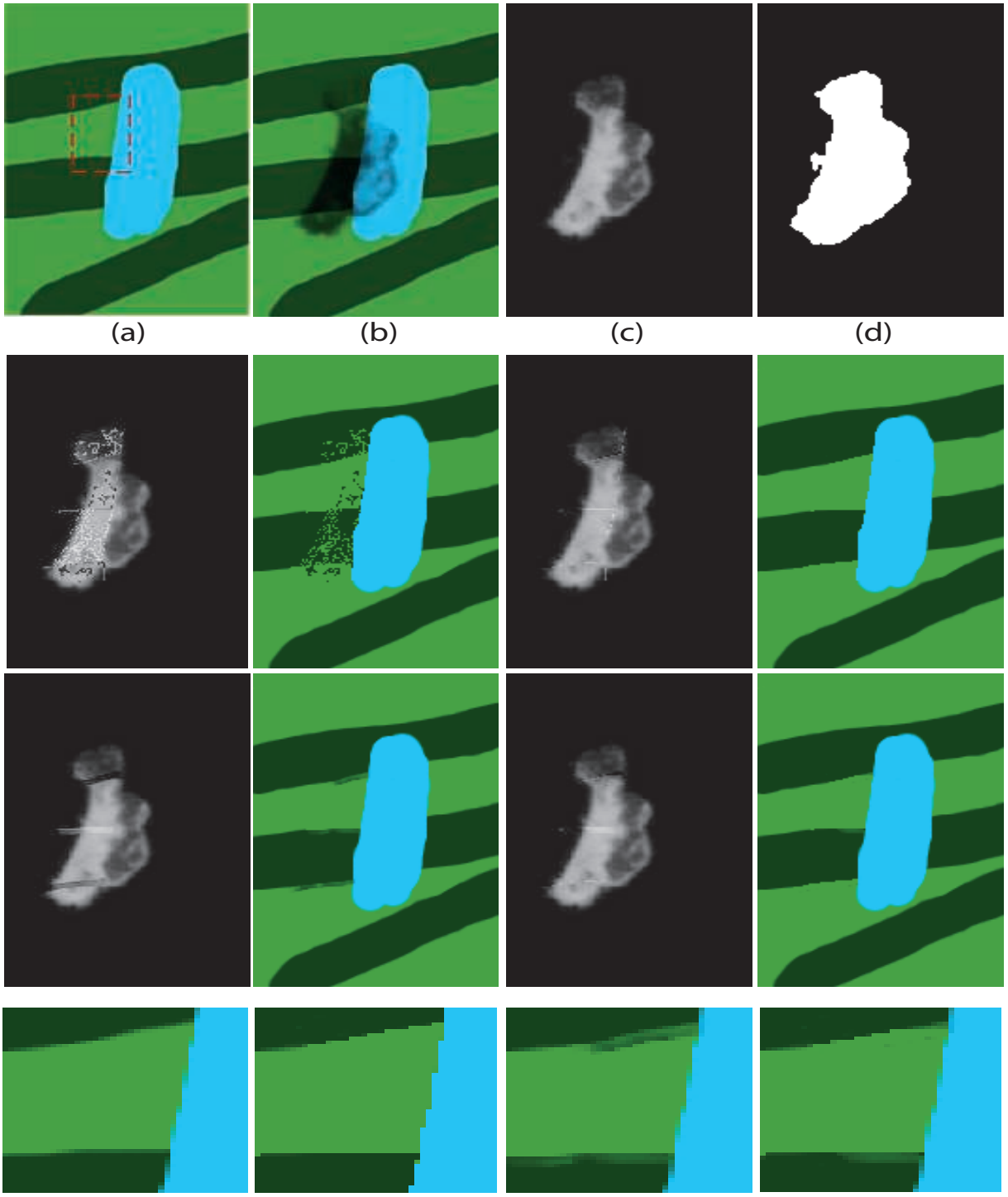


Figure 4.1: *First row: Illustrating the proposed corruption model. From left: Original image, corrupted image, a synthetic corruption matte α and the corruption mask $d(\cdot)$. Second row: matte and restoration using 'Bayesian Blotch Matting' and 'BTBR-S'. Third row: matte and restoration using BTBR-T and Spatio-temporal Fusion. Last row: zoomed area of the original image (a) (shown in red) and its spatial, temporal and spatio-temporal reconstructions respectively. BTBR-S generated sharp reconstruction of the blue edge while BTBR-T generated poor reconstruction of the green edges. 'Spatio-temporal Fusion' (BTBR-F) was able to compensate between those two artifacts.*

Here M_u denote the frame under examination. The likelihood $P(M_u|\alpha, B, F)$ forces the estimated $[\alpha, B]$ to cause $\alpha F + (1-\alpha)B$ to be close to the observed image M_u. The prior $P(\alpha, B|M_0, ...M_K, \alpha^{\mathcal{N}}, B^{\mathcal{N}}, F)$ enforces various smoothness constraints in space and time that cause the reconstructed patch to resemble the nearby clean data.

4.3 Maximum Likelihood Estimate

By considering the process as a matting exercise, we derive our first algorithm called *Bayesian Blotch Matting* i.e. BBM. Following Chuang et. al. [17], the clean image prior $P(B|M_n)$ is modeled as a mixture of Gaussians. That mixture is estimated from the nearby clean data in the examined frame. This forces the reconstructed data to be consistent with these regions. Clean samples are collected by extending a circular patch $\mathcal{R}$ from the examined site until a minimum number of uncorrupted pels $\mathcal{W}_u$ (=100 in our case) is included. The patch is then segmented into $\mathcal{W}_c$ color clusters (4 in our case) using a color quantization algorithm [60]. This yields a mixture of $\mathcal{W}_c = 4$ Gaussians, each with mean and co-variance $[\bar{\mathbf{B}}_\mathbf{j}, \mathbf{R}_{B_j}]$. This color segmentation step is necessary in order to capture the richness of the clean data.

Given observation noise $\sim \mathcal{N}(0, \sigma_e^2)$ in the compositing/observation model of equation 4.2 the likelihood w.r.t. the jth color cluster is then expressed as follows

$$P(M_u|\alpha, B, F) \propto \exp - \left(\frac{M_u - \alpha F + (1-\alpha)B}{2\sigma_e^2} + (B - \bar{\mathbf{B}}_\mathbf{j})^T \mathbf{R}_{B_j}^{-1} (B - \bar{\mathbf{B}}_\mathbf{j}) \right) \tag{4.4}$$

We use $\sigma_e^2 = 1$ for simplicity. The first term ensures that the estimated $[\alpha, B]$ reassemble the observed image M_u. The last term in the expression constrains the image data to obey a particular Gaussian color. This can also be thought of as a prior on that color. But because our color prior is in a sense data driven, we lump the two terms together in this likelihood expression.

4.3.1 Solving for $[\alpha, B]$

Given four Gaussians in the mixture model, attempting to solve for $[\alpha, B]$ using all four at once would lead in a sense to an *average* color constraint. Instead, we follow Chuang and choose to solve for $[\alpha, B]$ using each color component separately. This then yields four candidate solutions $[\alpha_j, B_j]$ and the candidate/component combination that maximizes the likelihood in equation 4.4 is selected as the solution at the examined pixel site.

We generate an estimate for B_j given α_j by substituting the likelihood into Equation 4.3, taking logarithms and differentiating wrt B_j in each color component separately. Given B_j we do the same for estimating α_j. This is the same optimization scheme used for deriving the MAP estimate of Bayesian Matting (see Appendix A). However the problem here is simpler as the foreground layer is already known ($F = 0$). Given $[B_j, M, F]$ are all 3-color vectors, the four

equations to solve are as follows.

$$\left[\sigma_e^2 \mathbf{R}_{B_j}^{-1} + I(1-\alpha_j)^2\right] B_j = \left[\sigma_e^2 \mathbf{R}_{B_j}^{-1} \bar{\mathbf{B}}_\mathbf{j} + (1-\alpha_j)M\right] \tag{4.5}$$

$$\alpha_j = \frac{(M-B_j)^T(F-B_j)}{||F-B_j||^2} \tag{4.6}$$

Here I is the 3×3 Identity matrix. The $[\alpha_j, B_j]$ pair for the background cluster j is calculated by iterating between equation 4.5 and equation 4.6 using the mean of the previously calculated opacity values as an initial α estimate. Performing this optimization for each color cluster produces a set of $\mathcal{W}_c$ $[\alpha, B]$ candidates for each site. The candidate producing the highest likelihood is then selected as the correct solution.

Figure 4.1 (second row, left) shows the generated corruption matte and the correction of Figure 4.1(b) using this approach. Here, figure 4.1(d) shows the used corruption mask. As shown the extracted mattes and original data are close to the ground-truth estimates of figure 4.1(a) and (c). However the generated results are noisy due to the absence of a spatial smoothness prior on the estimated $[\alpha, I]$.

Figure 4.2 shows the effect of using different $(\mathcal{W}_u, \mathcal{W}_c)$ configurations on the generated results. A small $\mathcal{W}_c$ value could fail to capture the richness of the clean data. The result could be an averaging like effect in the reconstructed original data (see figure 4.2 (c)). A small W_u value could fail to locate the correct original color if a large portion of this color is corrupted. This will reduce the chance of estimating the correct original color in the reconstructed regions (see figure 4.2 (d)).

BBM usually produces noisy results. This is because the maximum likelihood estimate does not guarantee the selection of the correct original color. This problem is solved by imposing spatial smoothness on the generated reconstruction as discussed next.

4.4 Spatial Priors

We now modify *Bayesian Blotch Matting* by imposing Spatial smoothness on $[\alpha, B]$. We call this the *Bayesian Transparent Blotch Remover - Spatial* algorithm i.e. BTBR-S. We use 8-connected MRFs in the usual way as follows:

$$P(\alpha|\alpha^{\mathcal{N}}) \propto \exp - \left(\beta_a \sum\nolimits_{k\in\mathcal{N}} \lambda_k |\alpha - \alpha_k|^2\right) \tag{4.7}$$

$$P(B|B^{\mathcal{N}}) \propto \exp - \left(\beta_b \sum\nolimits_{k\in\mathcal{N}} \lambda_k ||B - B_k||^2\right) \tag{4.8}$$

Here (β_a, β_b) are weights which configure the importance of each energy term while λ are parameters representing the level of correlation between adjacent pels in the original image. We use the crude assumption that $\lambda = 1$ at all sites. Even though crude, results (see Chapter 5) show accurate reconstruction of the original data even at textured regions.

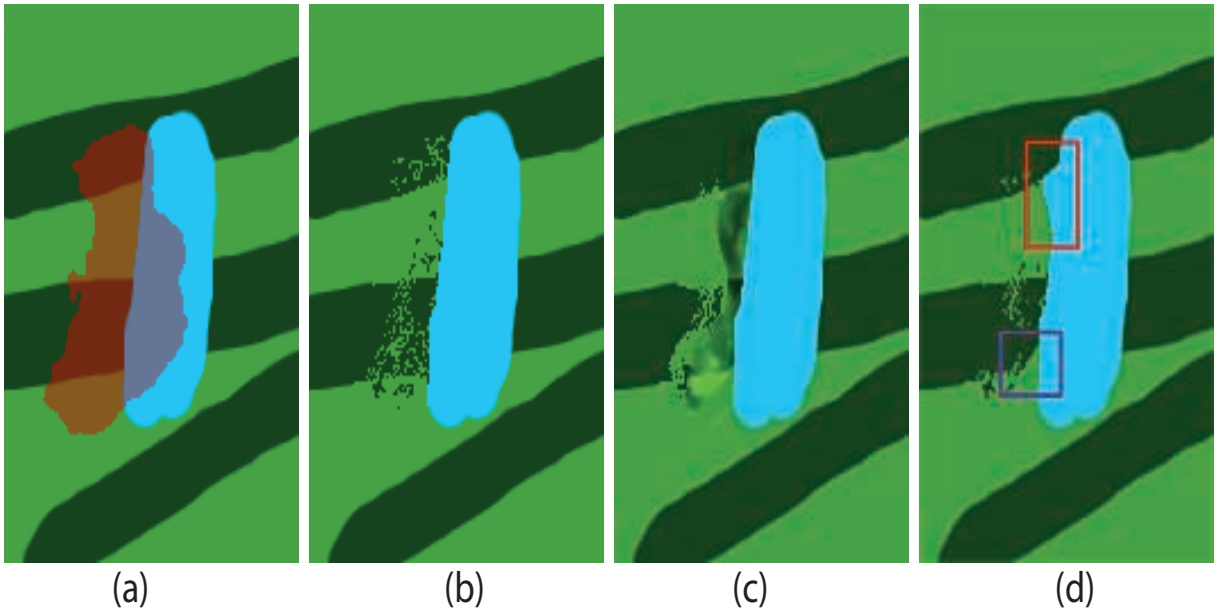


Figure 4.2: *From left: Corruption mask in red, BBM reconstruction with* $[\mathcal{W}_u, \mathcal{W}_c] = [100, 4], [100, 2], [20, 4]$ *respectively. In comparison with (b); (c) small* $\mathcal{W}_c$ *could fail to capture texture richness. The result is an averaging like effect in the generated reconstruction. (d) small* $\mathcal{W}_u$ *could fail in locating the correct original color if a large portion of this color is obscured. In the red rectangle shown, the light blue leaked into the light green as most of the light green is obscured by the corruption and so was not located due to the small* $\mathcal{W}_u$ *value. A similar effect is shown by the purple rectangle. Here the light green leaked into dark green as most of the dark green is obscured and so was not located due to the small* $\mathcal{W}_u$ *value.*

4.4.1 MAP Estimation With Spatial Priors

The posterior is optimized over two stages as follows.

1. For each segmented color cluster, its corresponding $[\alpha, B]$ estimate is calculated by iterating between equation 4.5 and equation 4.6 using 0.5 as an initial opacity value. This iterative process is performed until the absolute difference between the current likelihood and the previous likelihood is small enough. A value of 0.1 is used in this work. Performing this iterative process for each pel will generate a set of possible $[\alpha, B]$ candidates for each site.

2. The correct $[\alpha, B]$ candidate for each site is selected by finding the MAP estimate. This is done by choosing between two candidates at a time using QPBO Graph-Cut [69]. The winning candidate is then processed with the next solution candidate. This process is iterated with the remaining candidates until all candidates are considered (see figure 4.3).

Figure 4.4(a-b) shows the generated corruption matte and the restoration of Figure 4.1(b) using this approach (BTBR-S) with $[\beta_a, \beta_b] = [20, 0]$. As shown, an emphasis on the opacity

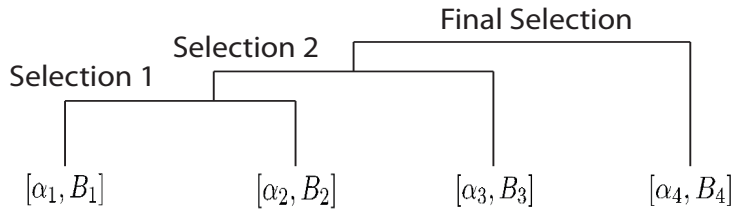


Figure 4.3: *The process of selecting the optimal* $[\alpha, B]$ *from a set of possible candidates. Two candidates are processed at a time via QPBO Graph-Cut and the candidate optimizing the MAP solution is selected.*

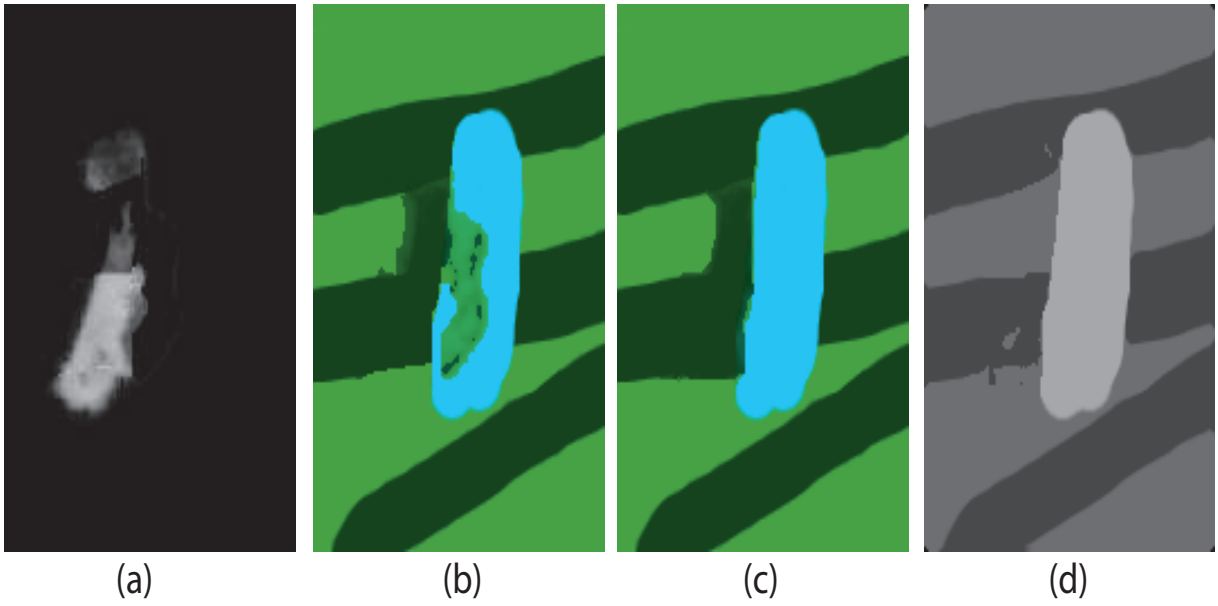


Figure 4.4: *(a-b) Generated matte and background restoration of figure 4.1(b) using 'BTBR-S' with* $[\beta_a, \beta_b] = [20, 0]$*. (c) Background restoration of figure 4.1(b) using 'BTBR-S' but with* $[\beta_a, \beta_b] = [0, 1]$*. As shown, different configurations of* $[\beta_a, \beta_b]$ *lead to different restorations. (d) Median filtered image of the background reconstruction of 'Bayesian Blotch Matting'* B*. In here, a* 5×5 *median filter is applied on* $\hat{B} = \frac{B_r + B_g + B_b}{3}$*, where* $[B_r, B_g, B_b]$ *are the red, green and blue channels of* B*. This image is used to infer the texture complexity of the original image* B*.*

smoothness could lead to matte oversmoothness. The visual impact could be severe reconstruction errors as shown in figure 4.4(b). Furthermore, figure 4.4(c) shows the generated background reconstruction of figure 4.1(b) with $[\beta_a, \beta_b] = [0, 1]$. As shown, an emphasis on the background smoothness could cause regions to bleed into its neighbors. To overcome reconstruction errors due to inaccurate settings of these parameters, two different configurations of $[\beta_a, \beta_b]$ are used, these being $[0.01, 0]$ and $[0, 1]$. The first configuration imposes spatial smoothness on the generated mattes while the other emphasizes background smoothness. The examined frame is

divided into 16×16 blocks and blocks with very low texture are assigned the configuration of $[\beta_a, \beta_b] = [0, 1]$ while the rest is assigned $[0.01, 0]$. This turns off background smoothness at textured regions to prevent possible bleeding of regions into their neighbors. To segment the examined frame into regions of strong and weak texture, we do the following. The 'Bayesian Blotch Matting' reconstruction of the examined frame B is transformed into $\hat{B} = \frac{B_r+B_g+B_b}{3}$ where $[B_r, B_g, B_b]$ are the red, green and blue components of B. $\hat{B}$ is then filtered by a 5×5 median filter. This generates a rough estimate of the underlying texture (see figure 4.4 (d)). Image gradients are calculated by applying the 'Roberts' edge detector on $\hat{B}$ where a pel is flagged as edge if its spatial derivative exceeds 3 gray scale levels. The texture complexity of each block is evaluated by calculating the L_0 norm of this edge map. A block is treated as textured if its L_0 norm value exceeds a threshold value of $\mathcal{T}$. High value of $\mathcal{T}$ may flag textured regions as untextured. This may cause reconstruction oversmoothness in textured regions as a high value of background smoothness will be assigned to these regions. To avoid this problem we use a small of value $\mathcal{T} = 2$. This value is fixed.

Figure 4.1 (second row, right) shows the generated matte and background reconstruction of figure 4.1(b) using 'BTBR-S'. In this example, the value of $[\beta_a, \beta_b] = [0.01, 0]$ is used over the whole image. As shown, 'BTBR-S' was able to eliminate the reconstruction noise generated in 'Bayesian Blotch Matting'. This is mainly due to imposing spatial smoothness on the generated results.

4.5 Temporal Priors

We can improve the background model $P(B|M_n)$ using information from nearby frames, especially given that the corruption does not occur in the same place in consecutive frames. This algorithm is called *Bayesian Transparent Blotch Remover - Temporal* i.e. BTBR-T.

The obscured original data in the current frame are estimated from previous and next frames using a simple block matching search with a block size configured to include at least 100 uncorrupted pels. For successful block matching the chosen block size should encompass texture richness of the examined neighborhood. Hence the block size is related to W_u and is therefore set to its value. This process is made robust to corruption by weighting out its effect with the opacity values of the maximum likelihood solution of 'BTBR-S'. The result is $\hat{M}_n$, a bi-directional motion compensation of the current frame at the corrupted sites. The background prior $P(B|\hat{M}_n)$ is then calculated using clean samples from a 3×3 $\hat{M}_n$ block that is centered at the examined site. The prior is modeled as one multivariate normal distribution $\mathcal{N}(\bar{\mathbf{B}}, \mathbf{R_B})$ having the mean and covariance of the clean samples. This prior forces the reconstructed data to be temporally consistent with the clean data in the nearby frames. Furthermore, the small block size imposes spatial smoothness on the generated parameters as the content of these blocks usually vary smoothly from one site to the other.

4.5.1 MAP Estimation With Temporal Priors

The same optimization scheme of Bayesian Blotch Matting is used. However here there is only one background color cluster.

Figure 4.1 (third row, left) shows the generated corruption matte and the restoration of figure 4.1(b) using this approach (BTBR-T). A slightly shifted image of the clean frame (by 2 pels vertically) is used as an estimate of $\hat{M}_n$. As shown, the new prior $P(B|\hat{M}_n)$ was able to restore a good portion of the original data with no reconstruction noise as in 'Bayesian Blotch Matting'.

4.6 MAP Estimate Through Spatio-temporal Fusion

The quality of the temporal solution degrades as motion gets more complicated while the quality of the spatial solution degrades as texture becomes more complex. Figure 4.1 (last row) outlines this fact by comparing different reconstructions of the red region in Figure 4.1(a). As shown, BTBR-S could lead to sharp edge reconstructions while BTBR-T often generates errors at regions of high Displaced Frame Difference (around green edges in this example). This calls for the generation of a final solution having minimum spatial and temporal reconstruction errors.

A spatio-temporal solution is generated by selecting between the spatial and temporal solutions by including a prior favoring either temporal or spatial $[\alpha, B]$ candidates. Here we use the QPBO Graph Cut as in BTBR-S. We call this the *Bayesian Transparent Blotch Remover - Fusion* algorithm i.e. BTBR-F. To choose between the two solutions we alter the probability $P(\alpha, B|\cdot)$ as follows

$$P(\alpha, B|M_0, ...M_K, \alpha^{\mathcal{N}}, B^{\mathcal{N}}, F) \propto P(B)P(\alpha|\alpha^{\mathcal{N}})P(B|B^{\mathcal{N}}) \tag{4.9}$$

where

$$P(B) \propto \begin{cases} \exp - \left((|DFD| - Q)^2/2\right) & \text{for the spatial candidate} \\ \exp -\beta_t \left(|DFD|^2/2\right) & \text{for the temporal candidate} \end{cases} \tag{4.10}$$

Here $P(B)$ is a prior introduced to bias the solution toward a temporal solution when there is good motion compensation between frames as measured by the DFDs. Q is a constant that can be related to motion complexity and which is set to 30 grey scale levels. This constant has the effect of favoring the temporal solution at sites of small motion errors where $DFD < \frac{Q}{2}$, while favoring the spatial solution otherwise. In addition, β_t is a parameter to configure the effect of the temporal solution on the final reconstruction. β_t is set to 1 in all frames except in frames containing pathological motion. Such frames have large motion errors and hence the temporal solution gets discarded from the final reconstruction by setting β_t to ∞. In the examined sequences we detect pathological motion manually. Alternatively, pathological motion can be automatically detected using the Corrigan et. al technique [19].

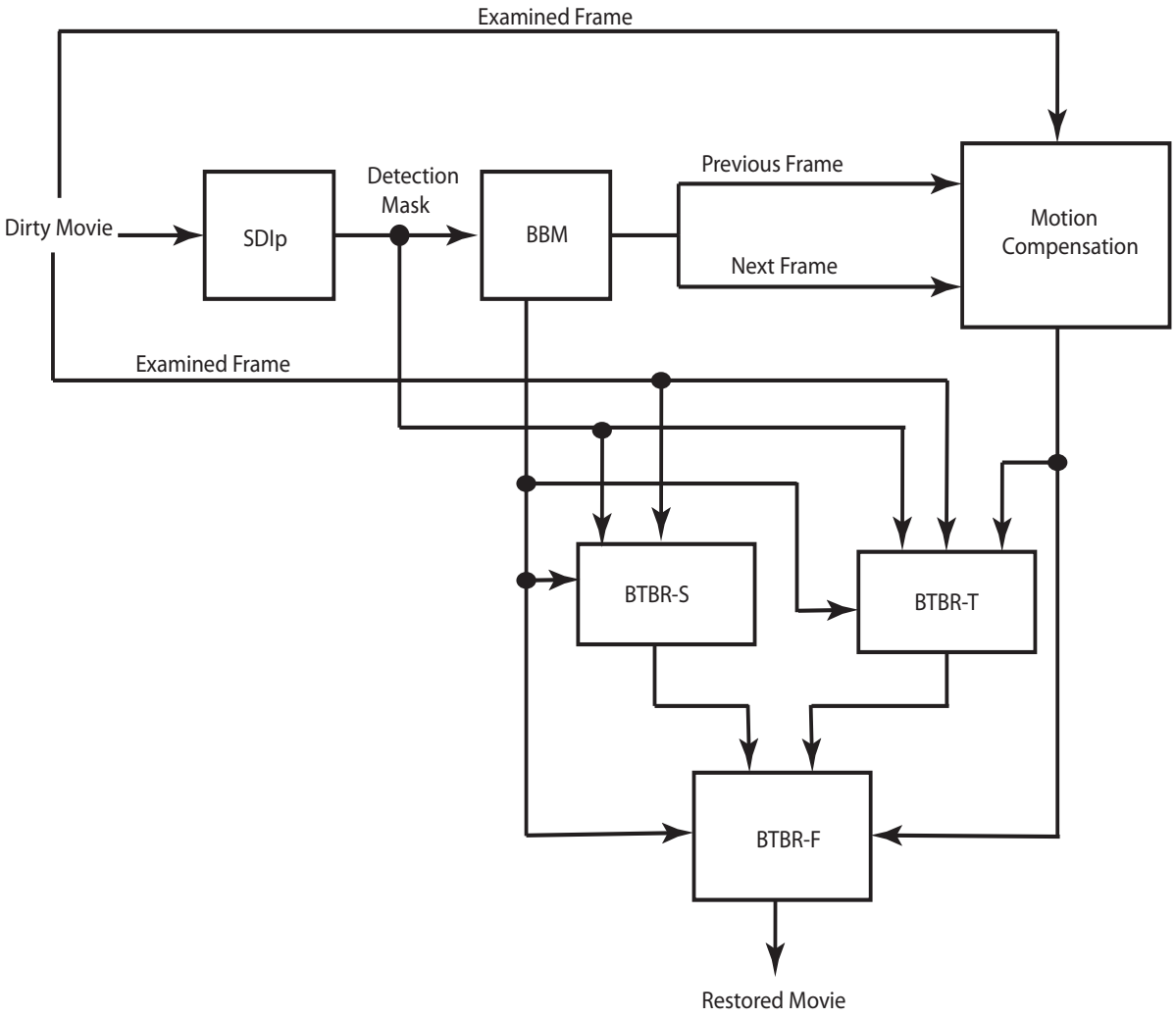


Figure 4.5: *Main Components of the BTBR System. Here BBM reconstructions are used to configure the* $[\beta_a, \beta_b]$ *values for BTBR-S, BTBR-T and BTBR-F. The motion compensated frames are used to configure* $P(B)$ *as shown by equation 4.10.*

Figure 4.1 (third row, right) shows the generated corruption matte and the restoration of Figure 4.1(b) using this approach with $[\beta_a, \beta_b] = [0.01, 0]$ and $\beta_t = 1$ for the whole image. As shown in Figure 4.1 (last row), Spatio-temporal BTBR-F was able to compensate between the errors generated in both the spatial and temporal reconstructions.

Figure. 4.5 shows the main components of our Blotch Removal System. We call it *Bayesian Transparent Blotch Remover* i.e. BTBR. Here the reconstructions of BBM are used to configure the opacity and background smoothness parameters $[\beta_a, \beta_b]$ of BTBR-S, BTBR-T and BTBR-F as discussed in Section. 4.4.1. Furthermore, the motion compensated frames are used to configure $P(B)$ as shown by equation 4.10.

4.7 Modifications for Line Removal

The background prior in BTBR-T is based on the assumption that corruption is temporally discontinuous and so the clean obscured data can be well estimated from nearby frames. This assumption is violated for line scratches as lines are temporally continuous events. Instead, for line removal we can build a new background prior from nearby reconstructed frames, assuming that at some point in the past and future the line does not appear in the original corrupted frames. We call this technique 'Bayesian Transparent Line Remover' i.e. BTLR

A line sequence starting and terminating with two line-free frames (M_0 and M_{N+1}) is therefore restored temporally over three stages (see figure 4.6)

1. A solution is generated by reconstructing the first corrupted frame using BTBR-T and propagating the reconstruction in the forward time direction. Reconstruction at $n-1$ is used to estimate $\hat{M}_n$ and the first corrupted frame is motion compensated using the line-free frame M_0. This will be referred to as 'Recursive Forward Reconstruction'.

2. A similar solution is generated but by starting the reconstruction from the last corrupted frame and propagating the solution in the backward direction. The main difference here is that $\hat{M}_n$ is estimated from the reconstruction at time $n+1$ and that the last corrupted frame M_N is motion compensated using M_{N+1}. The resulting reconstruction will be referred to as 'Recursive Backward-time Reconstruction'.

3. An overall temporal reconstruction is generated by fusing the Forward and Backward Reconstructions using the QPBO Graph Cut as in BTBR-S. In here background smoothness is emphasized relative to opacity smoothness as the small line width will prevent background oversmoothness. The resulting reconstruction will be referred to as 'Bi-directional-time Reconstruction'.

Figure 4.6 illustrates the proposed temporal line removal algorithm. As shown each method will accumulate reconstruction errors in the direction of propagation. However, bidirectional-time fusion is designed to minimize these errors in all frames. A final spatio-temporal solution is then generated by fusing the temporal and the spatial solutions using the same framework of section 4.6. For examined sequences we perform temporal propagation over (at maximum) 30 frames. Increasing the number of frames is expected to improve reconstruction because in theory objects would then have more time to move away from the degradation. However, this comes with an increased chance of motion estimation errors. Therefore this is a downfall which can only be addressed in the field as it works. In fact as our algorithm operates offline, we can take advantage of as many frames as is available.

Figure. 4.7 shows the main components of our Line Removal System. We call it *Bayesian Transparent Line Remover* i.e. BTLR. The forward motion compensated frames are used to configure the weights $P(B)$ in the final spatio-temporal fusion step (discussed in section 4.6). The stages of BBM, BTBR-S and BTBR-F used in BTLR are the same ones used in BTBR.

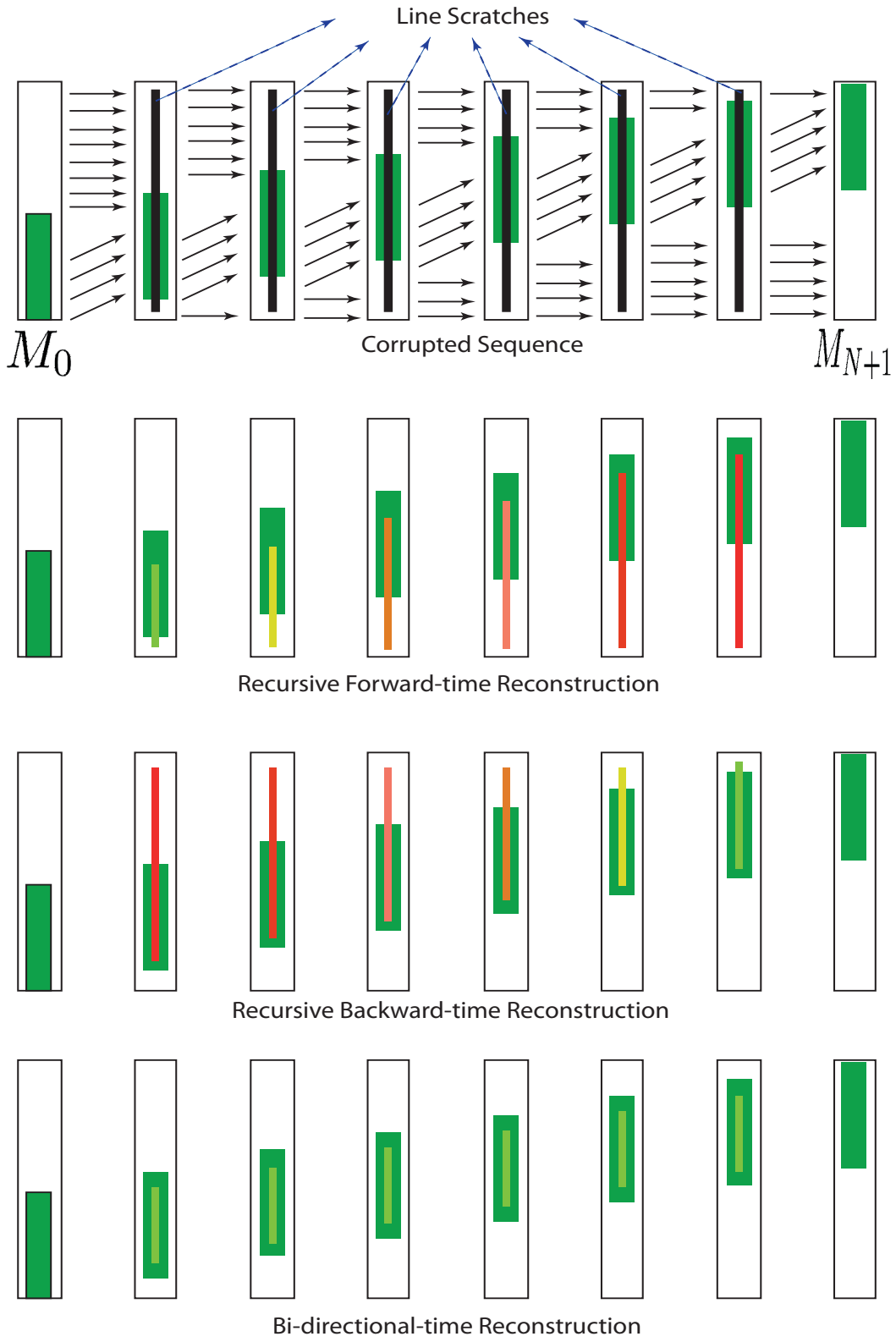


Figure 4.6: *Illustrating temporal restoration of a line sequence containing N corrupted frames. A green object is moving through time and estimated motion vectors are shown in black arrows. Reconstruction errors are represented with different degrees of 'red' ranging from 'light' for small errors to 'dark' for large errors. As shown, bidirectional fusion is designed to minimize reconstruction errors in all frames.*

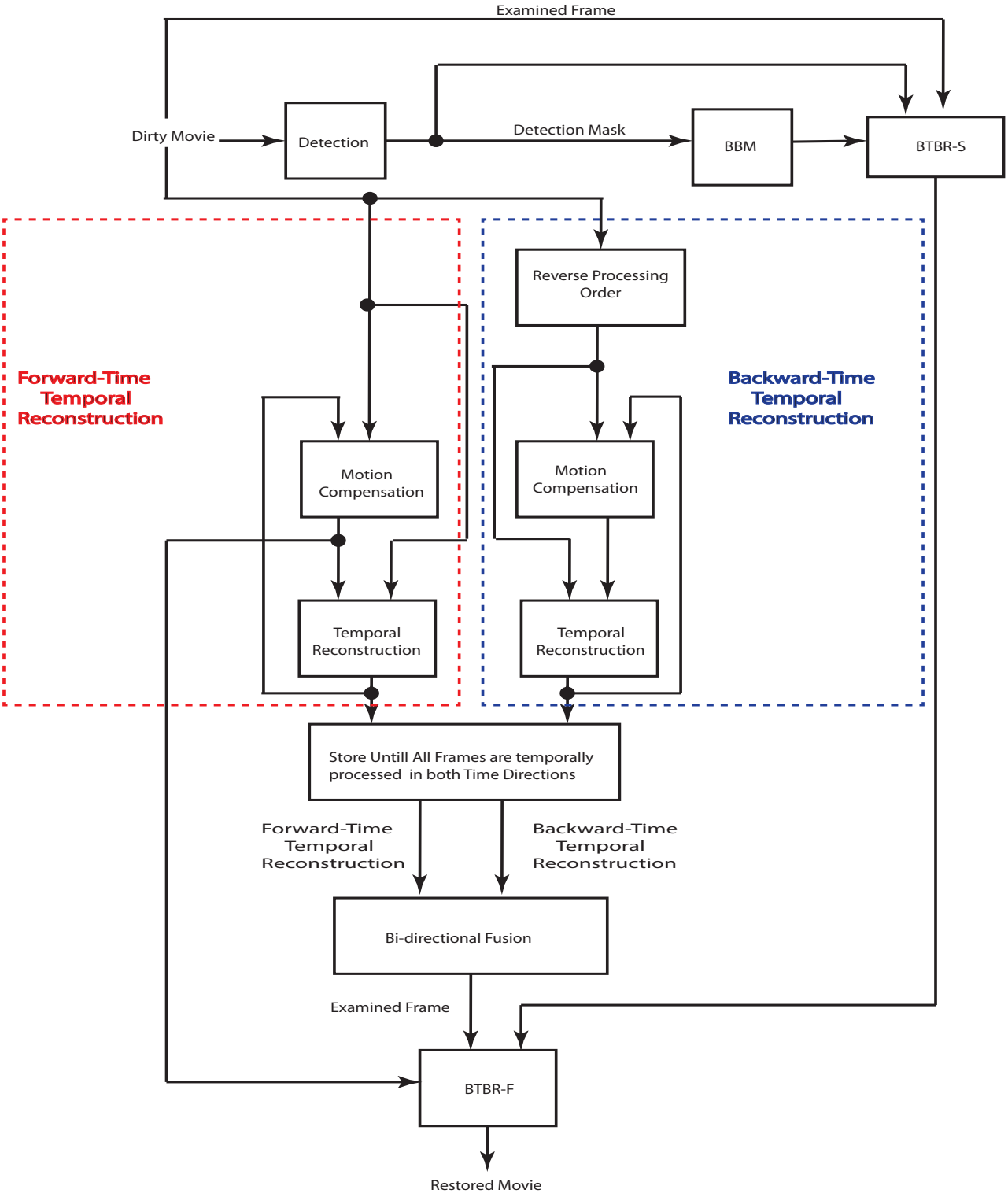


Figure 4.7: *Main Components of the BTLR System. The forward motion compensated frames are used to configure $P(B)$ as shown by equation 4.10. The block named 'Reverse Processing Order' in the blue region starts processing frame M_N down to M_1.*

4.8 Conclusion

This chapter has presented a new framework for removing dirt and lines from image sequences. It addresses the issue of correcting uncorrupted regions when traditional blotch removers are used and also allows the removal of semi-transparent damage in general from film sequences. The novelty here is in using a semi-transparent corruption layer. Corruption removal is then addressed as the problem of separating the corruption layer from the original background through a variant of the matting problem. A Bayesian framework was presented exploiting both spatial and temporal priors. Spatial and temporal information are used in a way such that restoration quality is maintained despite motion and texture complexity. The algorithm is initialized with rough binary corruption masks which are refined into a non-binary opacity mattes. These mattes estimate the amount of dirt at each pixel and therefore discard clean regions from the correction process.

5 Infrared Analysis

This chapter shows experiments used to evaluate our blotch (BTBR) and line (BTLR) removal techniques. Results are evaluated by comparing the reconstructed corruption opacities and reconstructed original data against ground truth estimates. Ground truth estimates are derived from IR scans of corruptions. IR scans are traditionally used for corruption detection [23] and hence they are commonly generated by some archive film holders e.g. Institut national de l'audiovisuel . Our ground-truth generation technique first estimates ground-truth corruption opacities by projecting the IR scans on to the corruption opacity space. Here we derive a transformation which relates the IR scans grayscale values with corruption opacities. Ground-truth estimates of the original data are then calculated by weighting out the effect of the corruption using the estimated ground-truth opacities. Comparisons against ground-truth estimates show that our corruption removal technique estimates corruption borders with much better accuracy than current removal techniques. This avoids correcting uncorrupted sites.

This chapter starts by outlining the experiments used in evaluating our corruption removal techniques. We then show how to exploit IR film scans for ground-truth estimates. We then use the ground-truth estimates to evaluate our corruption removal techniques against real and synthetic data.

5.1 Experimental Procedures

Test Sequences: Five standard definition (720×576) sequences are used in evaluating the performance of the blotch removal processes: LabB1, ArtB1, DanceB1, DanceB2 and LadyDollB1

with 100, 40, 100, 60 and 100. frames respectively. Most sequences undergo fast motion and contain moderate texture. However LadyDollB1 contains moderate texture and undergoes slow motion. Examples of frames from these sequences are shown in figure 5.1- 5.2. The IR scans of the sequences are shown on the right. LabB1 is created by corrupting a clean sequence B using the relation $M(\mathbf{x}) = (1 - \alpha(\mathbf{x}))B(\mathbf{x})$. Here α is the dirt opacity obtained from IR scans. The process of getting the dirt opacity from the IR scans is discussed in details in section 5.2. The other four sequences ArtB1, DanceB1, DanceB2 and LadyDollB1 show real archived footage containing blotches. Similarly, four line sequences are used to evaluate the performance of line scratch removal: LabL1, DanceL1, DanceL2 and DanceL3, with 25, 25, 18 and 15 frames respectively. Again these show fast motion and contain moderate texture. LabL1 shows synthetic line scratches created in the same way as LabB1 and the others contain real line scratches. Examples of frames from DanceB1 and LabL1 are shown in figure 5.3.

Examined Algorithms: There are a large variety of blotch remover processes that have been proposed, hence we compare our removal techniques with the JONDI estimator of Kokaram et al [47], since that is the most general of the frameworks proposed in the past and it is the basis of many commercial blotch removal processes. For line removal we compare the results against JOMBEI which is in a sense a spatial version of JONDI for line removal [46]. We also compare with one commercially available software suite called Furnace from The Foundry [2]. This uses an implementation of ML3Dex [51] for removal. To illustrate the importance of the opacity term in generating accurate reconstructions, we compare the results against an implementation of BTBR which removes the effect of the opacity term i.e. α is replaced by a binary index. Here the image compositing term (the first term) of the BBM likelihood is ignored (see equation 4.4). This can be regarded as an approximation of Efros et al. texture synthesis technique [24, 25]. We call this implementation BTBR-AOFF. We also compare the results of BTBR and BTLR against BBM.

To show the importance of using both spatial and temporal information in our restoration technique, we compare the results of BTBR-F against BTBR-S and BTBR-T. Even though BTBR does not address the problem of detection in an explicit form, the estimation of α is a detection refinement step. We therefore show how the estimated α refines the SDIp detections. We also examine how BTBR performs on grayscale sequences. Finally we discuss the computational load of our technique. To evaluate the original data reconstruction quality of the examined restoration techniques we use the SSIM measure of Wang et al. [93]. This measures the structural similarity between the generated reconstructions and the ground-truth estimates. The SSIM measure takes values from 0 to 1 where 1 denotes an exact match with the ground-truth estimate.

Ground truth Generation: Ground truth is traditionally hard to come by since it can only generated by painstaking hand painting of missing patches. In this case, with a model that considers semi-transparency that is even more difficult. However, we have acquired Infra Red (IR) scans of film material which yield ground truth on real degraded sequences. IR scans are

Figure 5.1: *From top: Frames from LabB1, DanceB1 and DanceB2 and their corresponding IR scans on the right. All Images except LabB1 are Courtesy of Institut national de l'audiovisuel.*

Figure 5.2: *From top: Frames from ArtB and LadyDollB1 and their corresponding IR scans on the right. Images Courtesy of Institut national de l'audiovisuel.*

dark in corrupted sites and bright in clean regions (see figure 5.1-5.2, right). We project the IR grayscale values into the corruption opacity domain α. This generates ground-truth opacity estimates. These estimates are used to create LabB1 and LabL1 by corrupting a clean sequence B using $M(\mathbf{x}) = (1 - \alpha(\mathbf{x}))B(\mathbf{x})$. Furthermore, ground-truth estimates of the original data are then estimated by weighting out the effect of the corruption using the ground-truth opacities. The next section discusses how ground truth is acquired in detail. We then go on to discuss algorithm performance by examining different aspects of our technique.

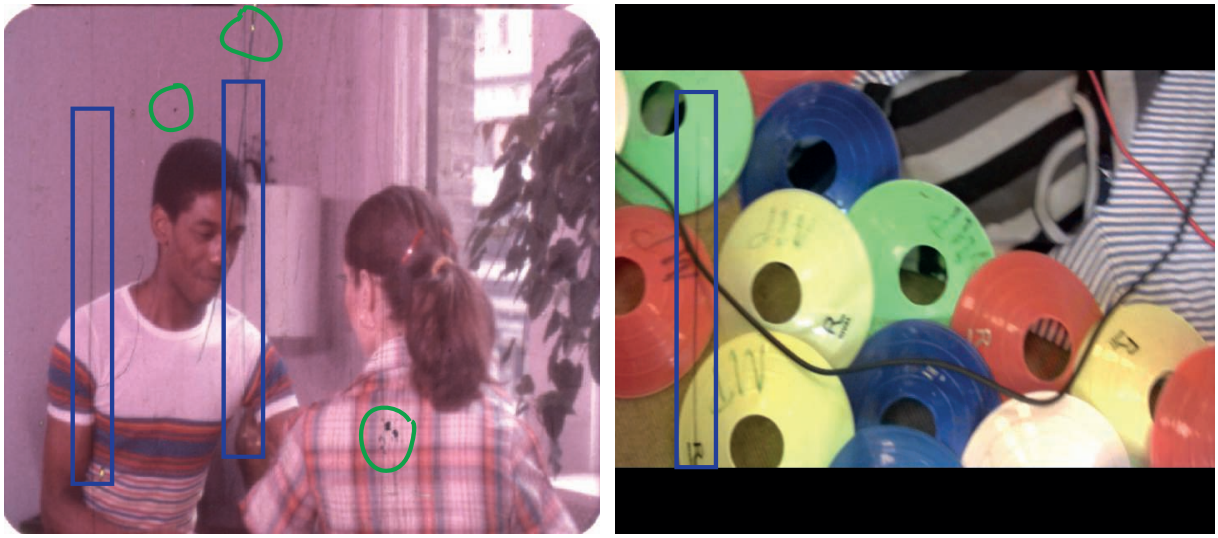


Figure 5.3: *Frames from DanceL1 (left) and LabL1 (right). Line scratches are shown in blue. DanceL1 contains blotches (shown in green) as well as lines.*

5.2 Groundtruth from IR

Corrupted sites are detected using a simple threshold operation on the IR scans of the examined sequences. The two real sequences Art and Dance are examined where Dance contains DanceB1, DanceB2, DanceL1, DanceL2 and DanceL3. Threshold values of 210 and 180 are used for the Art and Dance sequences respectively, where a pel is flagged as corrupted if it falls below the threshold. This thresholding operation yields the ground truth binary corruption mask d_τ for each sequence. We extract the ground-truth opacities α_τ for each sequence by relating the actual grayscale value of the IR to the estimated opacities. Here we only examine sites denoted by d_τ. Corruption opacities are estimated using BTBR and figure 5.4 shows the IR/Corruption opacity plot for the Art and Dance sequences. IR scans are spread over the range $0-1$ for the simplicity of illustration where 1 denotes a highly corrupted region. Three different fitting functions are superimposed: $y = x^n$, $y = a.x^2 + b.x + c$ and $y = \gamma.exp(\lambda x) + k$. The fitting error ϵ for a specific fitting function is defined as follows

$$\epsilon = \sqrt{\frac{\sum_{i=1}^{M} \sum_{j=1}^{N_i} (\alpha_i^f - \alpha_{i,j})^2}{\sum_{i=1}^{M} N_i}} \tag{5.1}$$

Here i indexes the IR range $0-1$ with a step of 0.01, α_i^f is the i^{th} fitted value and $\alpha_{i,j}$ is the j^{th} BTBR blotch opacity at the i^{th} IR value. Table. 5.1 shows the fitting errors of the different fitting functions for each examined sequence. Table. 5.1 shows that the function $y = a.x^2 + b.x + c$ gives the best fit in both Art and Dance sequences and hence we use this function to transform the IR grayscale values to ground truth opacities.

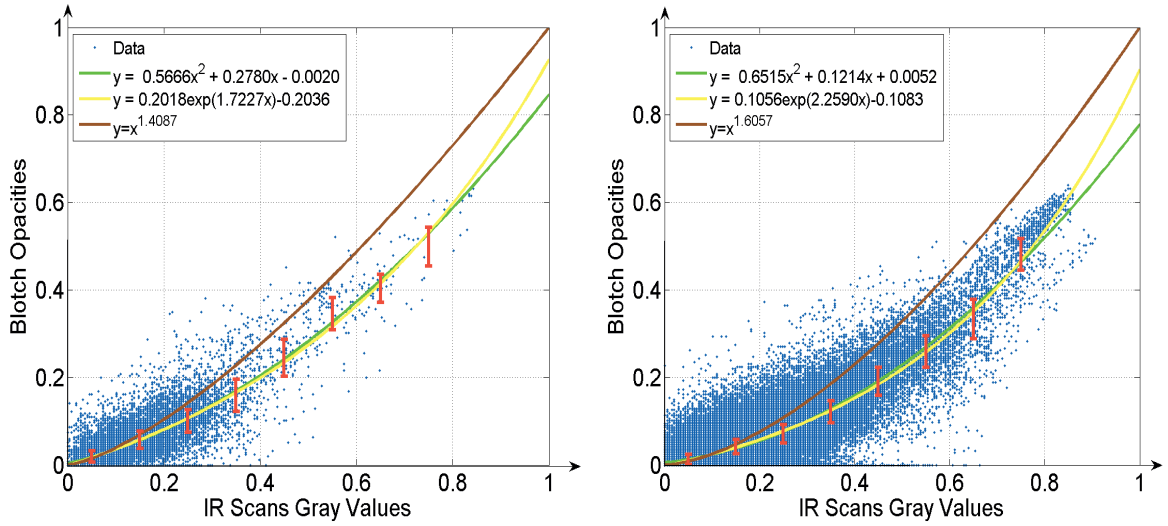


Figure 5.4: *Corruption opacities α vs. IR values for the Art (left) and Dance sequences (right). The red bars denote the standard deviations of α. Both $y = a.x^2 + b.x + c$ and $y = \gamma.exp(\lambda x) + k$ generate a good fit of the IR/Corruption opacity relation.*

		Art	Dance
$f_P = x^n$	n	1.4087	1.6057
	ϵ	0.0919	6.4773
$f_E = \gamma.exp(\lambda x) + k$	γ	0.2018	0.1056
	λ	1.7227	2.2590
	k	-0.2036	-0.1083
	ϵ	0.0076	0.0703
$f_Q = a.x^2 + b.x + c$	a	0.5666	0.6515
	b	0.2780	0.1214
	c	-0.0020	0.0052
	ϵ	0.0061	0.0833

Table 5.1: *Different functions for estimating the ground-truth corruption opacities (see f_P, f_E and f_Q respectively) and their corresponding lack of fit ϵ with the BTBR opacities. f_Q generates the least lack of fit for both examined sequences*

.

Figure 5.5-5.6 shows some blotches, their IR scans, the calculated dirt opacities using the derived IR/Corruption relation and the corresponding original data reconstruction. The function $y = a.x^2 + b.x + c$ is used as the IR/Corruption relation and reconstruction is achieved by inverting the effect of the dirt using the matting equation directly. More explicitly the ground-

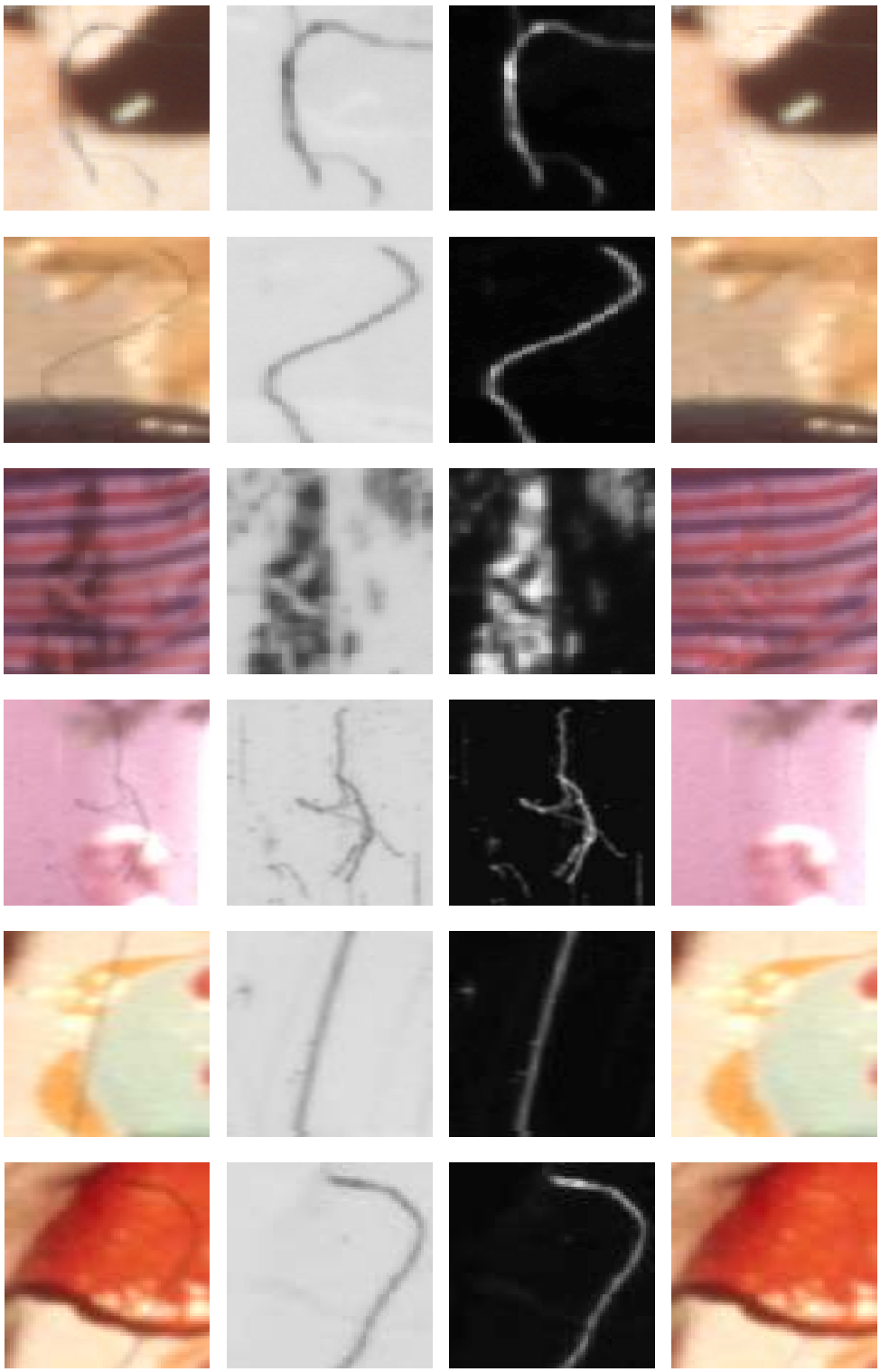


Figure 5.5: *Ground-truth opacities and reconstructions calculated using the method discussed in section 5.2. From left; Corrupted Image, corresponding IR scan, estimate ground-truth corruption opacity and the corresponding reconstruction. As shown the reconstruction successfully recovers the underlying original data.*

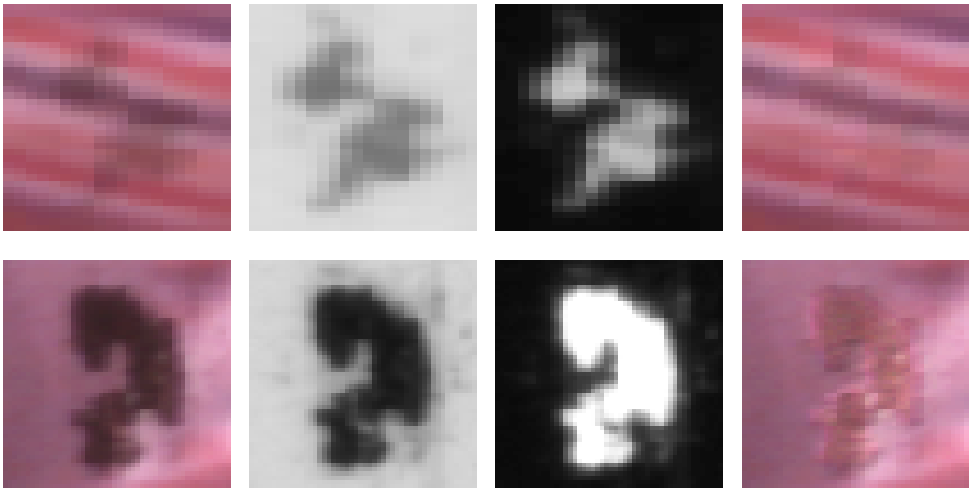


Figure 5.6: *Ground-truth opacities and reconstructions calculated using the method discussed in section 5.2. From left; Corrupted image, corresponding IR scan, estimate ground-truth corruption opacity and the corresponding reconstruction. The reconstruction successfully recovers the underlying original in the first row however it failed to remove highly opaque corruptions in the last row.*

truth estimate of the original data B_τ given the ground-truth opacities α_τ is calculated using

$$B_\tau(\mathbf{x}) = \frac{M(\mathbf{x})}{(1 - \alpha_\tau(\mathbf{x}))} \tag{5.2}$$

where $\mathbf{x}$ denote the examined pels respectively and M is the corrupted frame. As shown in figure 5.5-5.6 the ground-truth reconstructions in the last column reassemble the original data correctly in most examples. However, the quality of reconstruction degrades as corruptions become opaque. Figure 5.6 (last row) shows an example where the ground-truth reconstruction fails to fully remove very dark corruption. For very dark corruptions the numerator and denominator of equation 5.2 approach zero. As a result $B_\tau(\mathbf{x})$ approaches being undefined and hence the reconstruction becomes meaningless.

5.3 Reconstruction Comparison

5.3.1 Comparison against Current Techniques

Figure 5.7-5.8 shows the original data reconstruction quality of LabB1, DanceB1, DanceB2 and ArtB1 as generated by different blotch removers. Image sequence results can be found in the accompanying DVD. Five techniques are examined being BTBR, JONDI, BBM, Furnace and BTBR-AOFF. The graphs shows SSIM with ground-truth estimates for every examined frame. These results are summarized in table 5.2. As shown in table. 5.2, BTBR outperforms the other techniques in most of the examined sequences. The examined techniques remove blotches in most

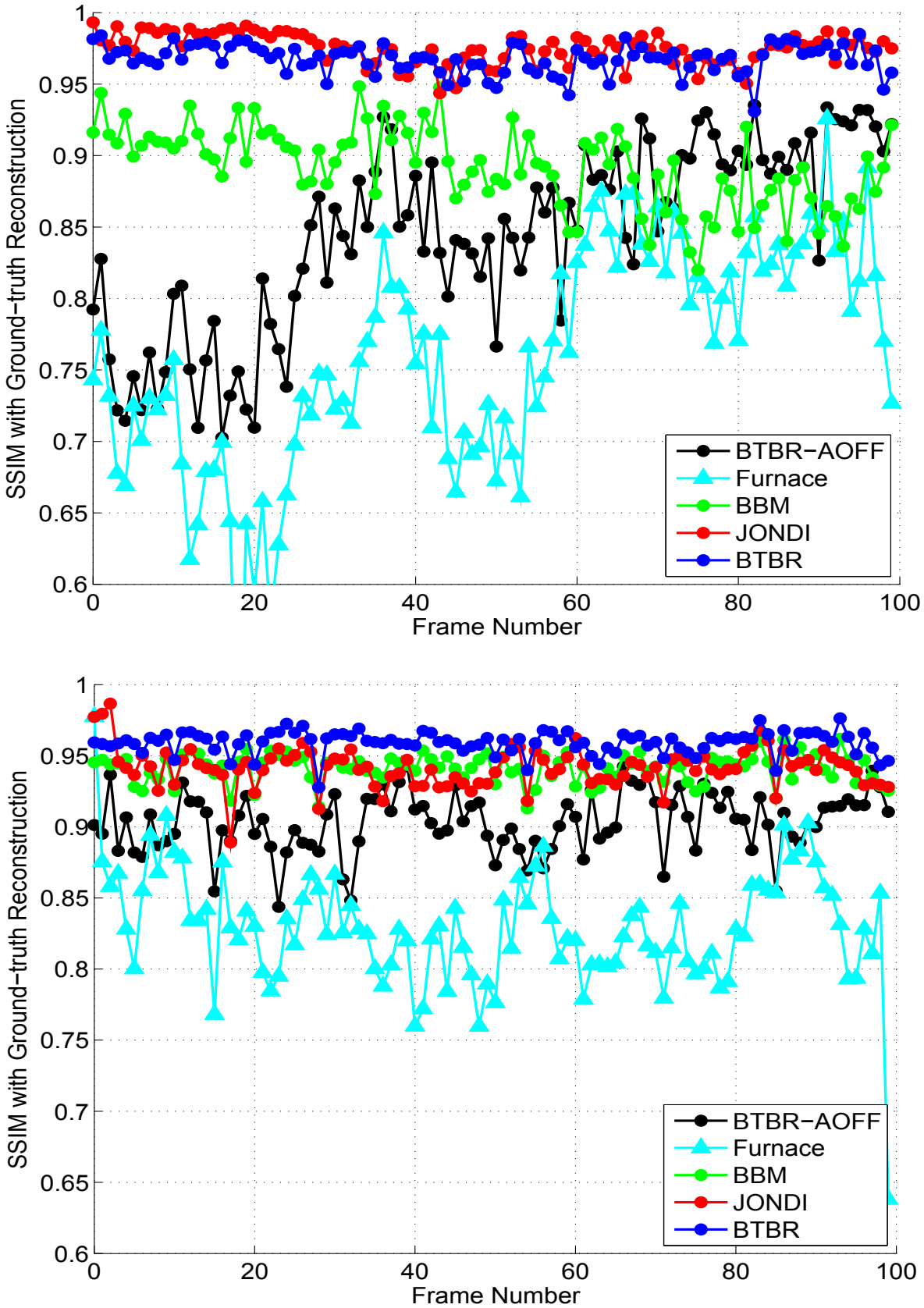


Figure 5.7: *Reconstruction quality of LabB1 (top) and DanceB1 (bottom) with different blotch removers. BTBR outperforms most of the other techniques.*

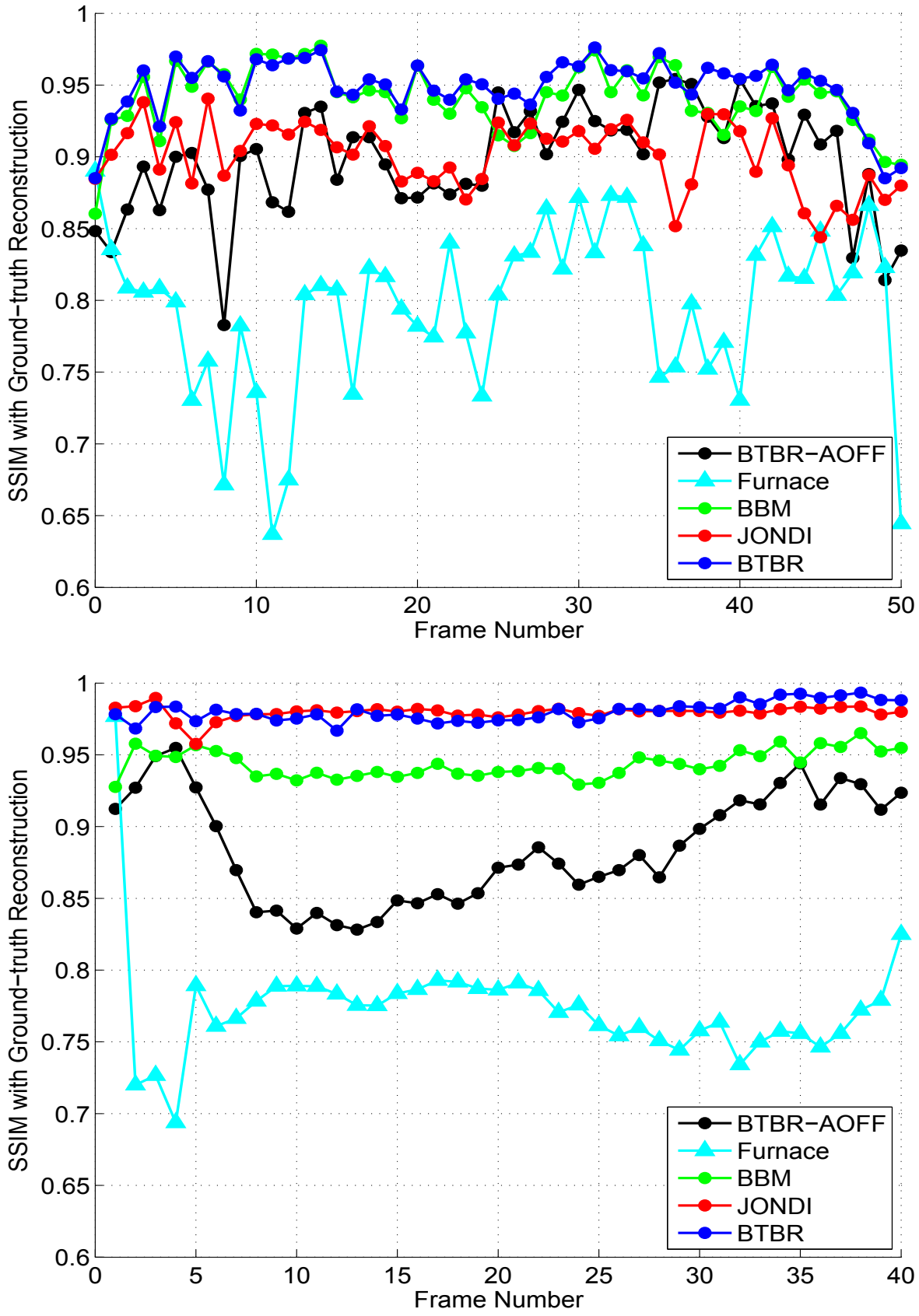


Figure 5.8: *Reconstruction quality of DanceB2 (top) and ArtB1 (bottom) with different blotch removers. BTBR outperforms most of the other techniques.*

	LabB1	DanceB1	DanceB2	ArtB1
Furnace	0.76±0.08 0.44 0.93	0.83±0.04 0.64 0.98	0.80±0.06 0.64 0.89	0.77±0.04 0.69 0.98
JONDI	0.98±0.01 0.94 0.99	0.94±0.01 0.89 0.99	0.90±0.02 0.84 0.94	0.98±0.00 0.96 0.99
BTBR-AOFF	0.84±0.07 0.70 0.94	0.90±0.02 0.84 0.94	0.90±0.04 0.78 0.95	0.88±0.04 0.82 0.95
BBM	0.89±0.03 0.82 0.95	0.94±0.01 0.91 0.96	0.94±0.02 0.86 0.98	0.94±0.01 0.93 0.97
BTBR	0.97±0.01 0.93 0.99	0.96±0.01 0.93 0.98	0.95±0.02 0.89 0.98	0.98±0.01 0.97 0.99

Table 5.2: *Reconstruction quality against ground truth for the examined blotch sequences. SSIM (Structural Similarity Measure of [93]) is used here where SSIM of 1 denotes reconstruction that is identical to the ground truth estimate. For each sequence the average, minimum and maximum SSIM are shown respectively. The average and minimum SSIM shows that BTBR is the best performing technique.*

of the processed frames however they generate errors in clean regions which are misclassified as corrupted. Furnace usually generates a temporal averaging effect in clean regions. This visual effect is clearly visible in all examined frames in figure 5.9-5.11 and figure 5.13-5.15 (second row, left). BBM often generates noisy reconstructions as it does not impose any smoothness on the generated results. This is shown in figure 5.10 and figure 5.13-5.15 (third row, left, shown in green). Using a corruption model with binary opacity values instead of non-binary values generates large restoration artifacts in clean regions as shown in figure 5.9-5.10 and figure 5.14-5.15 (second row, right, shown in red).

BTBR is more robust to complicated motions such as those arising due to pathological motion or fast motion. Such motions violate the motion translational model used by SDIp and hence generate large false SDIp detection rate. As JONDI treats corruption as an opaque layer it usually passes regions incorrectly classified as blotches to the correction stage. However, thanks to the semi-transparent corruption model we often discard false alarms from the correction stage. Figure 5.9-5.12 (shown in blue) shows examples where the examined regions are undergoing complicated motion and hence have a high false detection rate. JONDI generates clearly visible artifacts in these regions while our technique maintains the original information. In addition to

JONDI's failure in handling complicated motion, sometimes it misclassifies very dark corruptions as being clean and hence does not remove them. Figure 5.13 (third row, right, shown in blue) shows an example where JONDI misclassifies a dark corruption as being a clean region. The result here is incomplete blotch removal.

(a) Original (b) Detection

(c) Furnace (d) BTBR-AOFF

(e) BBM (f) JONDI

(g) BTBR

Figure 5.9: *In clockwise direction: Frame 67 of DanceB1, its SDIp mask and reconstruction using different techniques. Blotches are shown in black boxes. BTBR-AOFF and JONDI generate reconstruction errors in uncorrupted regions (shown in red and blue respectively).*

(a) Original (b) Detection

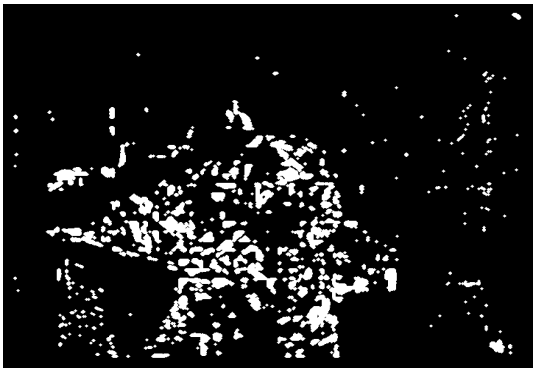
(c) Furnace (d) BTBR-AOFF

(e) BBM (f) JONDI

(g) BTBR

Figure 5.10: *In clockwise direction: Frame 76 of DanceB1, its SDIp mask and reconstruction using different techniques. Blotches are shown in black boxes. BBM generates noisy results (shown in green) while BTBR-AOFF and JONDI generate reconstruction errors in uncorrupted regions (shown in red and blue respectively).*

(a) Original



(b) Detection

(c) Furnace

(d) BTBR-AOFF

(e) BBM

(f) JONDI

(g) BTBR

Figure 5.11: *In clockwise direction: Frame 77 of DanceB1, its SDIp mask and reconstruction using different techniques. Blotches are shown in black boxes. JONDI generate reconstruction errors in regions misclassified as corrupted (shown in blue).*

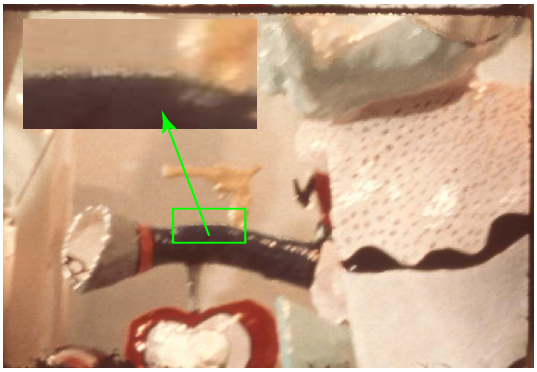
Figure 5.12: *From top; Frame 190 (left column) and 191 (right column) of DanceB2; SDIp masks, JONDI reconstruction, BTBR reconstruction. Blotches are shown in black boxes. JONDI generates large reconstruction errors in regions misclassified as corrupted (shown in blue).*

(a) Original

(b) Detection

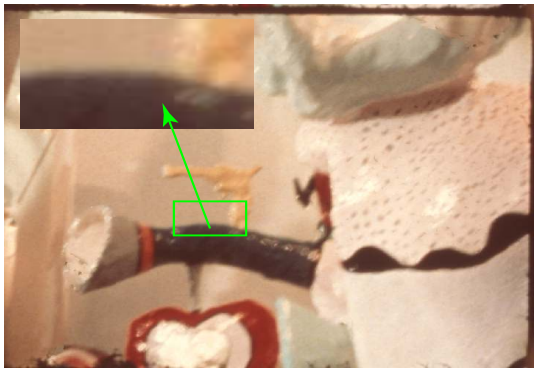
(c) Furnace

(d) BTBR-AOFF



(e) BBM

(f) JONDI



(g) BTBR

Figure 5.13: *In clockwise direction: Frame 32 of ArtB1, its SDIp mask and reconstruction using different techniques. Blotches are shown in black boxes. JONDI failed in removing a very dark corruption (shown in blue) while BBM generates noisy results (shown in green).*

(a) Original

(b) Detection

(c) Furnace

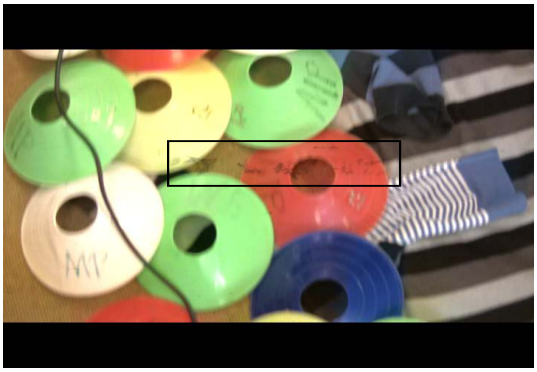
(d) BTBR-AOFF

(e) BBM

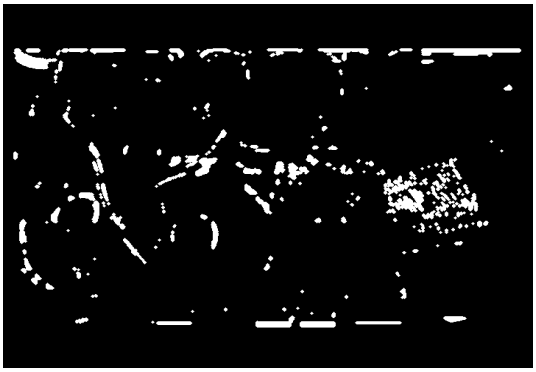
(f) JONDI

(g) BTBR

Figure 5.14: *In clockwise direction: Frame 28 of LabB1, its SDIp mask and reconstruction using different techniques. Blotches are shown in black boxes. BBM generates noisy results (shown in green) while BTBR-AOFF generates large reconstruction errors in uncorrupted regions (shown in red).*



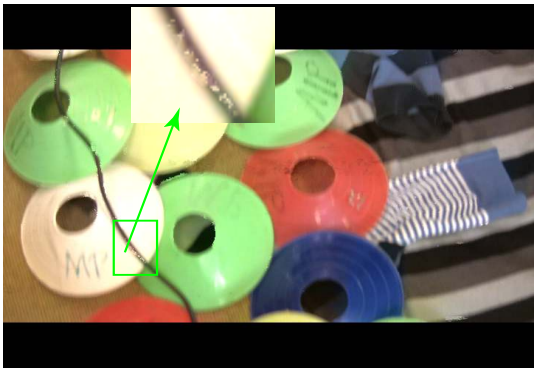
(a) Original



(b) Detection

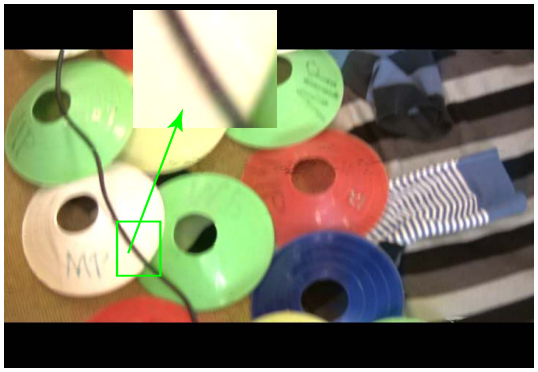
(c) Furnace

(d) BTBR-AOFF



(e) BBM

(f) JONDI



(g) BTBR

Figure 5.15: *In clockwise direction: Frame 64 of LabB1, its SDIp mask and reconstruction using different techniques. Blotches are shown in black boxes. BBM generates noisy results (shown in green) while BTBR-AOFF generates large reconstruction errors in regions misclassified as corrupted (shown in red).*

	LabL1	DanceL1	DanceL2	DanceL3
Furnace	0.92±0.04 0.85 0.97	0.85±0.05 0.75 0.92	0.88±0.03 0.83 0.93	0.92±0.03 0.85 0.95
JOMBEI	0.95±0.01 0.93 0.97	0.92±0.02 0.88 0.96	0.91±0.02 0.87 0.96	0.95±0.02 0.89 0.96
BTBR-AOFF	0.94±0.02 0.87 0.97	0.90±0.03 0.85 0.96	0.91±0.03 0.88 0.96	0.92±0.03 0.85 0.96
BBM	0.94±0.03 0.88 0.98	0.91±0.02 0.85 0.84	0.90±0.02 0.85 0.9	0.92±0.02 0.89 0.94
BTLR	0.96±0.02 0.93 0.98	0.92±0.02 0.87 0.96	0.94±0.02 0.90 0.96	0.95±0.02 0.90 0.96

Table 5.3: *Reconstruction quality against ground truth for the examined line sequences. SSIM [93] is used here where SSIM of 1 denotes reconstruction that is identical to the ground truth estimate. For each sequence the average, minimum and maximum SSIM are shown respectively. The average and minimum SSIM shows that BTLR is the best performing technique.*

Figure 5.16-5.17 shows the original data reconstruction quality of LabL1, DanceL1, DanceL2 and DanceL3 as generated by different line removers. Image sequence results can be found in the accompanying DVD. Five techniques are examined being BTLR, JOMBEI, BBM, Furnace and BTBR-AOFF. The graphs show SSIM with ground-truth estimates for every examined frame. These results are summarized in table 5.3. As shown in table 5.3, BTLR outperforms the other techniques in most of the examined sequences. Figure 5.18 shows the BTLR reconstruction of three consecutive frames from LabL1. The Line scratch (shown in blue) is removed successfully from the examined frames. Figure 5.19 zooms in on the first frame of figure 5.18 and compares the BTLR reconstruction with BTBR-AOFF, Furnace, JOMBEI and BBM. JOMBEI often blurs the corruption (see blue box A & C). BTBR-AOFF generates artifacts in clean regions due to the absence of a non-binary opacity term (box C & D). Furnace generates incomplete removal (box A) and BBM generates noisy reconstruction as expected (box B). Figure 5.20 shows three frames of LabL1 and its BBM and BTLR reconstruction. Its clear that BBM generates noisy results due to the absence of spatial reconstruction smoothness (see blue boxes). This noise manifests as flickering during video playback.

Figure 5.21 shows line and blotch removal from the Dance sequence using our corruption removal techniques. Blotches and lines are removed separately and both restorations are then

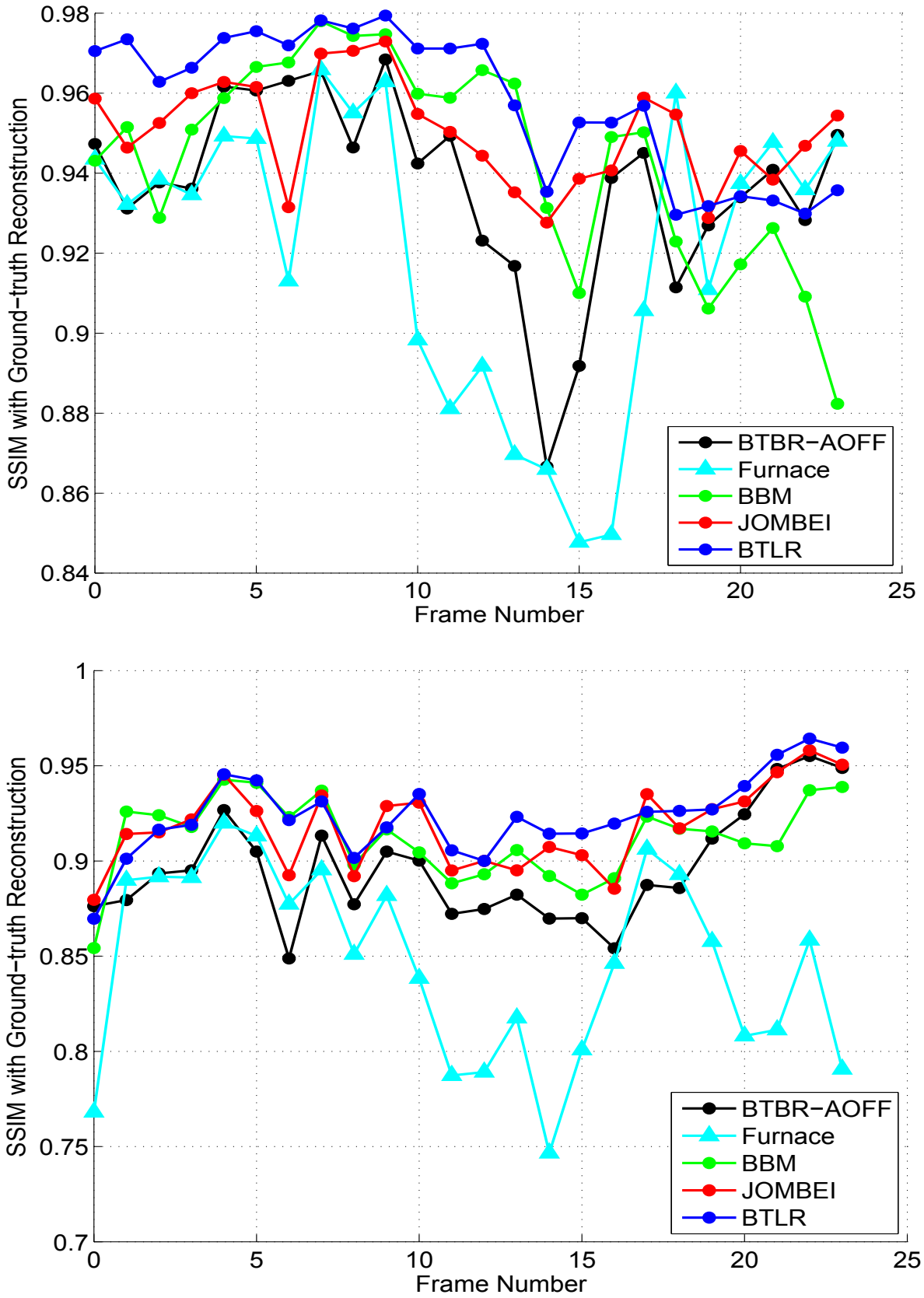


Figure 5.16: *Reconstruction quality of LabL1 (top) and DanceL1 (bottom) with different line removers. BTLR outperforms most of the other techniques in most of the examined frames.*

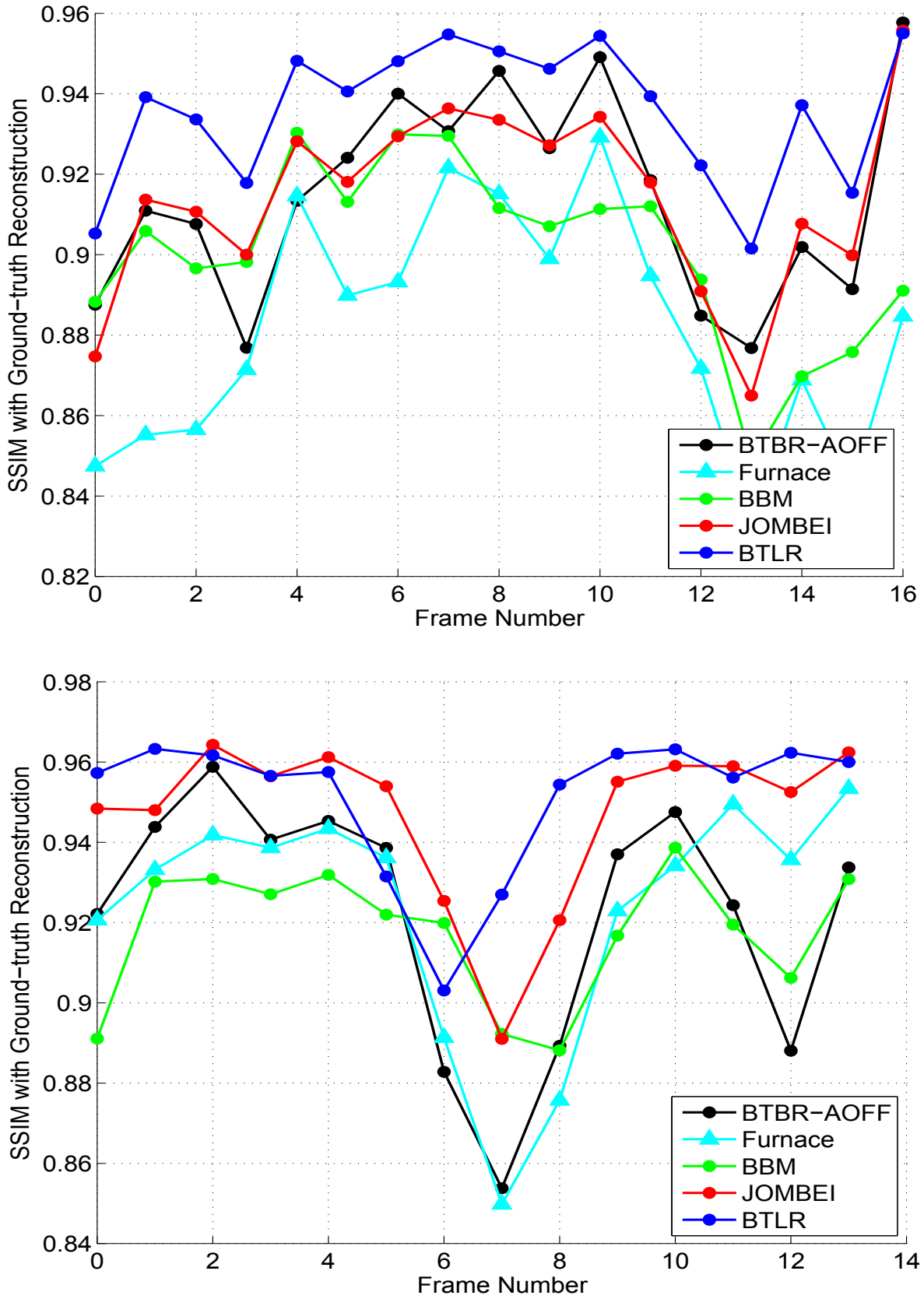


Figure 5.17: *Reconstruction quality of DanceL2 (top) and DanceL3 (bottom) with different line removers. BTLR outperforms most of the other techniques in most of the examined frames.*

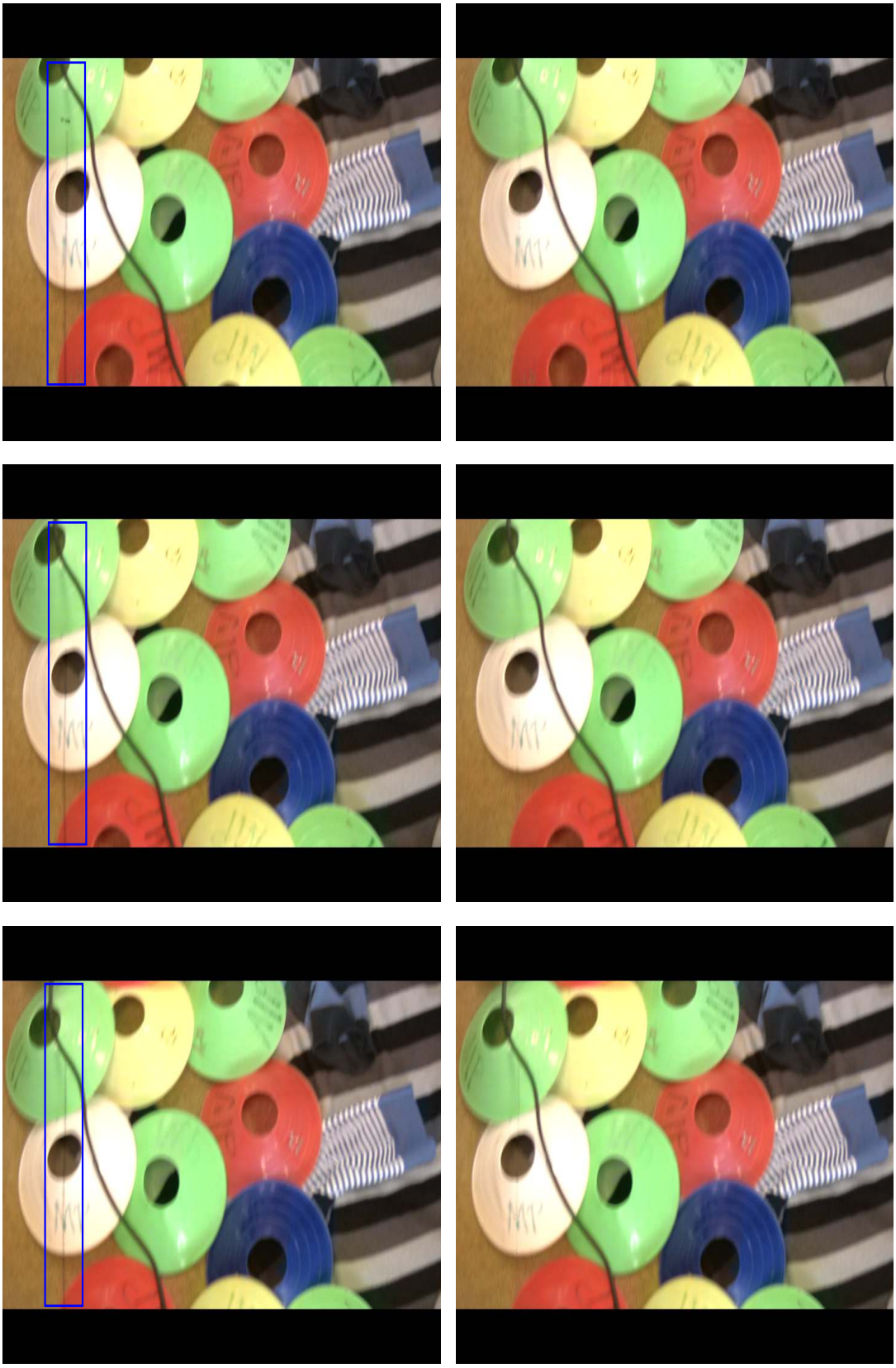


Figure 5.18: *From top: Frames 19, 20 and 21 from LabL1 (left column) and line removal results of BTLR on the right. Line scratches (shown in blue) are successfully removed by BTLR.*

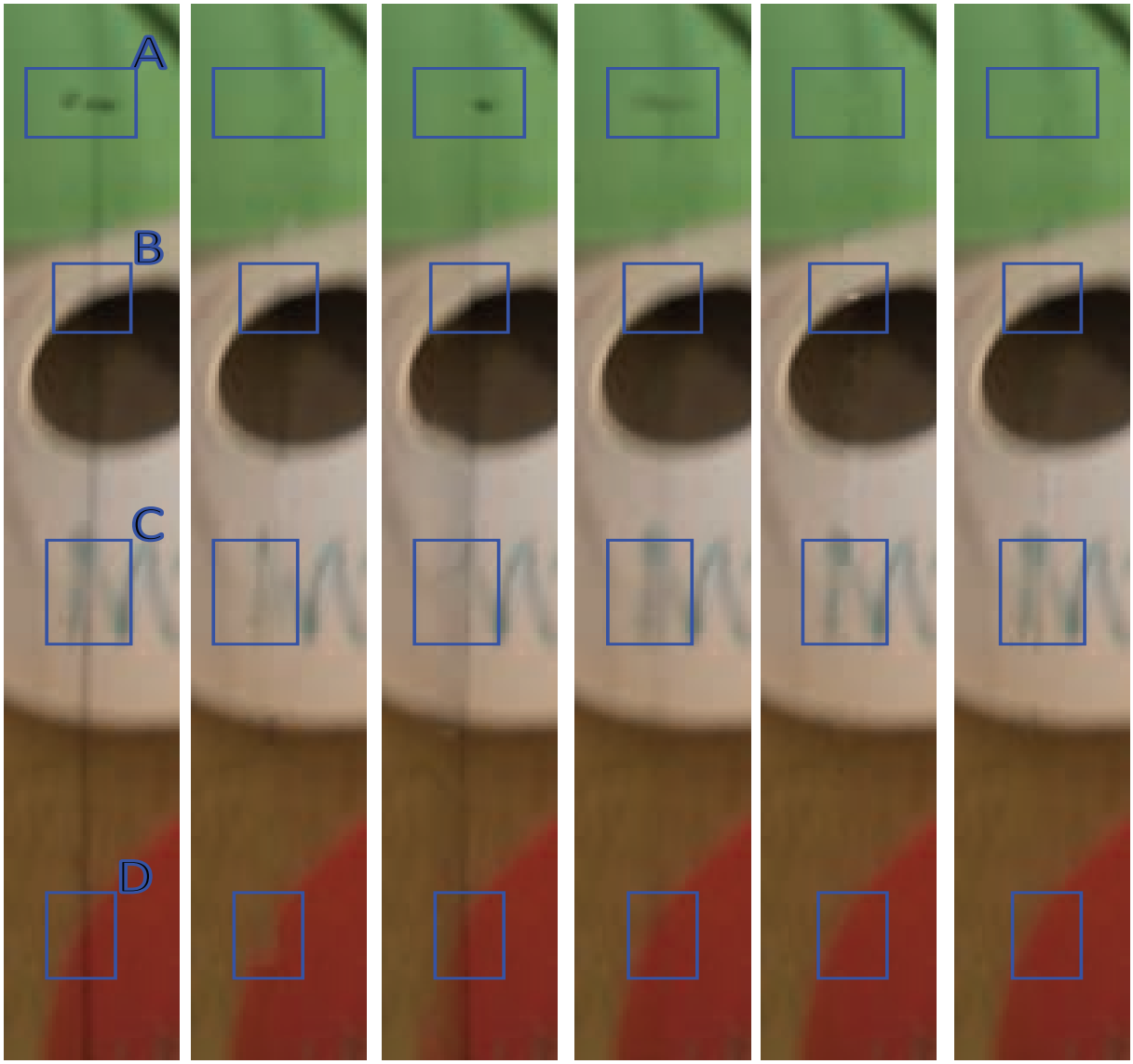


Figure 5.19: *From left; frame 19 of LabL1 and its restoration using BTBR-AOFF, Furnace, JOMBEI, BBM and BTLR respectively. BTBR-AOFF corrupts clean regions (box C & D), Furnace produces incomplete removal (box A), JOMBEI often blurs the corruption (box A & C) and BBM generates noisy reconstruction (box B). BTLR however generated the best restoration (compare the restorations shown in blue boxes).*

assembled to form the final restored sequence. The Dance sequence in figure 5.21 contains the line scratch of DanceL1 (shown in blue, left side of the corrupted frame) and the line scratch of DanceL2 (shown in blue, around the center of the corrupted frame). It also contains blotches from DanceB1 (shown in green). As shown in figure 5.21 our corruption removal techniques were able to remove both lines and blotches successfully.

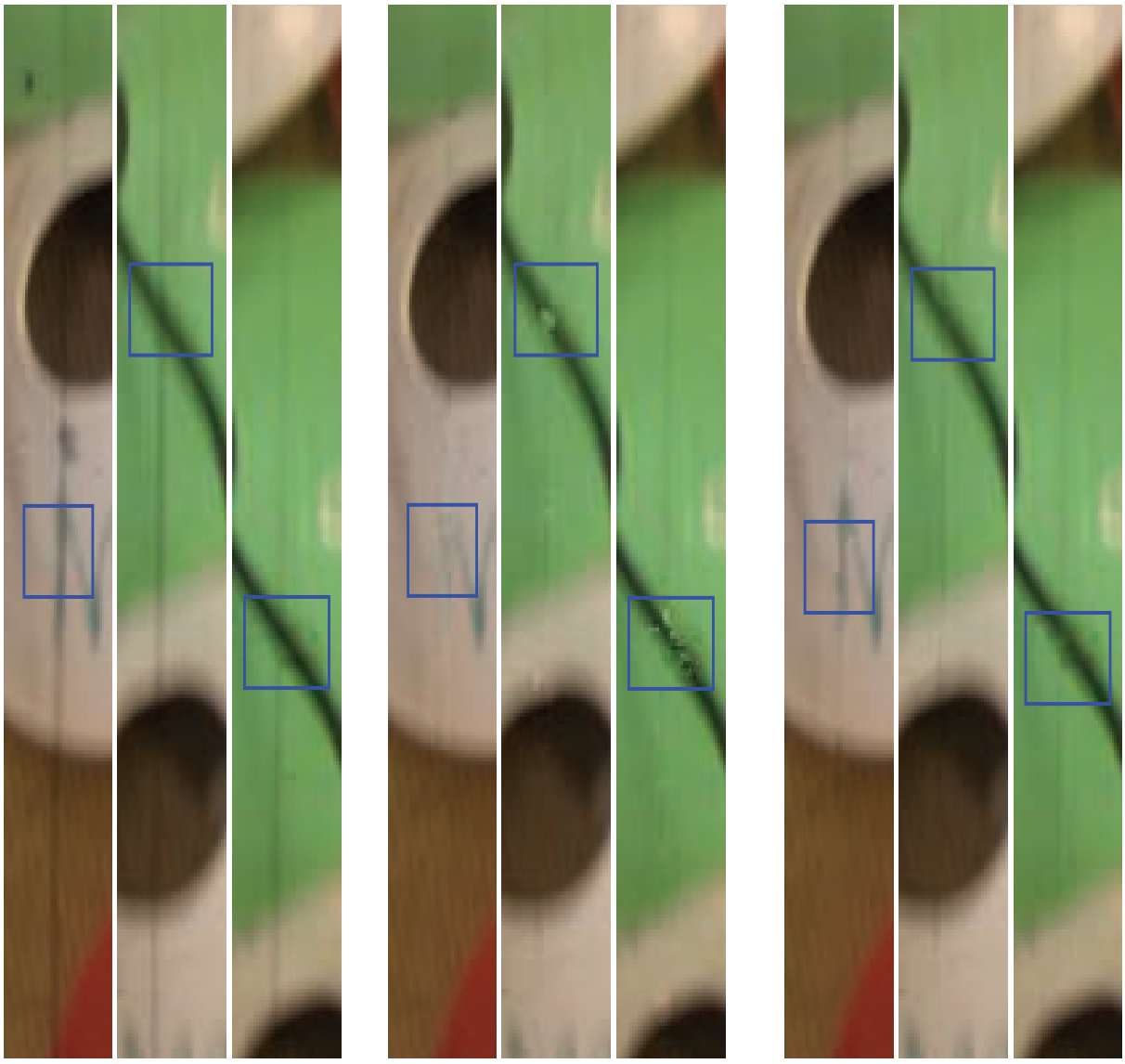


Figure 5.20: *From left; frame 18, 23 and 28 of LabL1 and its BBM and BTLR reconstruction respectively. BTLR often generates smoother results than BBM due to the incorporation of a reconstruction smoothness term (see blue boxes). The noisy restoration of BBM manifests as flickering during video playback.*

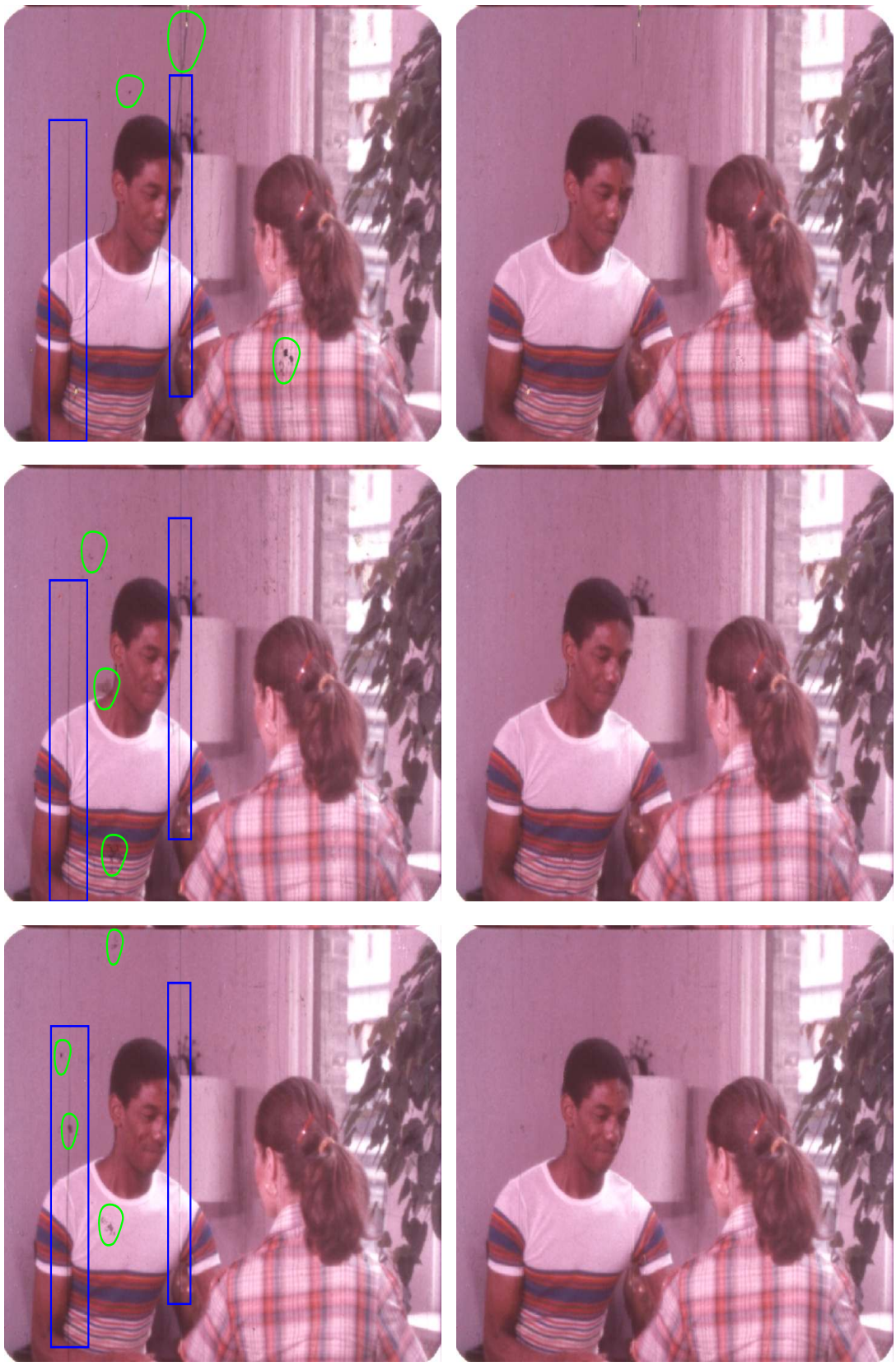


Figure 5.21: *Line and blotch removal from the Dance sequence using BTBR and BTLR. Lines and blotches are shown in the corrupted frames on the left column in blue and green respectively. Restoration results are on the right column.*

	LabdyDollB1	DanceB1
BTBR-S	0.89±0.04 0.97 0.77	0.95±0.01 0.92 0.97
BTBR-T	0.92±0.03 0.82 0.97	0.90±0.03 0.82 0.96
BTBR-F	0.94±0.03 0.82 0.98	0.96±0.01 0.93 0.98

Table 5.4: *Reconstruction quality against ground truth for DanceB1 and LadyDollB1. SSIM [93] is used here where an SSIM of 1 denotes reconstruction that is identical to the ground truth estimate. For each sequence the average, minimum and maximum SSIM are shown respectively. The average and minimum SSIM shows that BTBR-F fuses the spatial and temporal reconstructions in a way such that restoration is maintained despite motion and texture complexity.*

5.3.2 The Importance of Spatial and Temporal Information in BTBR

Figure 5.22 shows the original data reconstruction of DanceB1 and LadyDollB1 using BTBR-S, BTBR-T and BTBR-F. These results are summarized in table. 5.4. DanceB1 undergoes complicated motion. As a result the temporal restoration (BTBR-T) generates more errors than the spatial restoration (BTBR-S). Here BTBR-F favors the spatial restoration of BTBR-S over the temporal restoration of BTBR-T. Examples of temporal reconstruction errors are shown in blue in figure 5.23-5.24. LadyDollB1 contains simple motion between frames and therefore its temporal restoration generates fewer errors than the spatial restoration (see table. 5.4 and figure 5.22). Hence this time BTBR-F favors the restoration of BTBR-T over the restoration of BTBR-S. Example of spatial restoration failure is shown in figure 5.25 in blue. The BTBR-F results of DanceB1 and LadyDollB1 show that our corruption removal technique BTBR uses both spatial and temporal information in a way such that restoration is maintained despite motion and texture complexity.

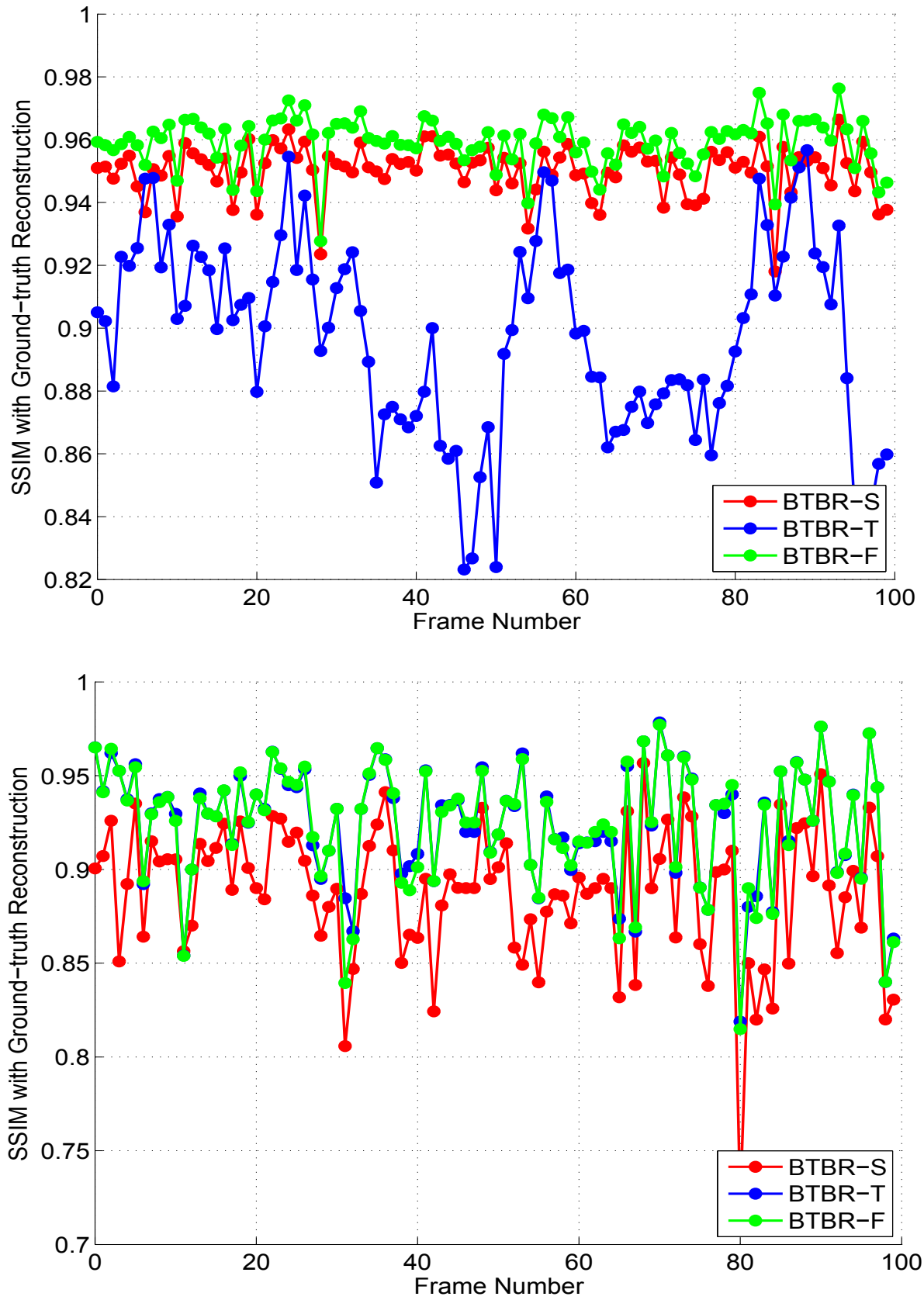


Figure 5.22: *Reconstruction quality of DanceB1 (top) and LadyDollB1 (bottom) as generated by BTBR-S, BTBR-T and BTBR-F. BTBR-F fuses the spatial and temporal reconstructions in a way such that restoration is maintained despite motion and texture complexity.*

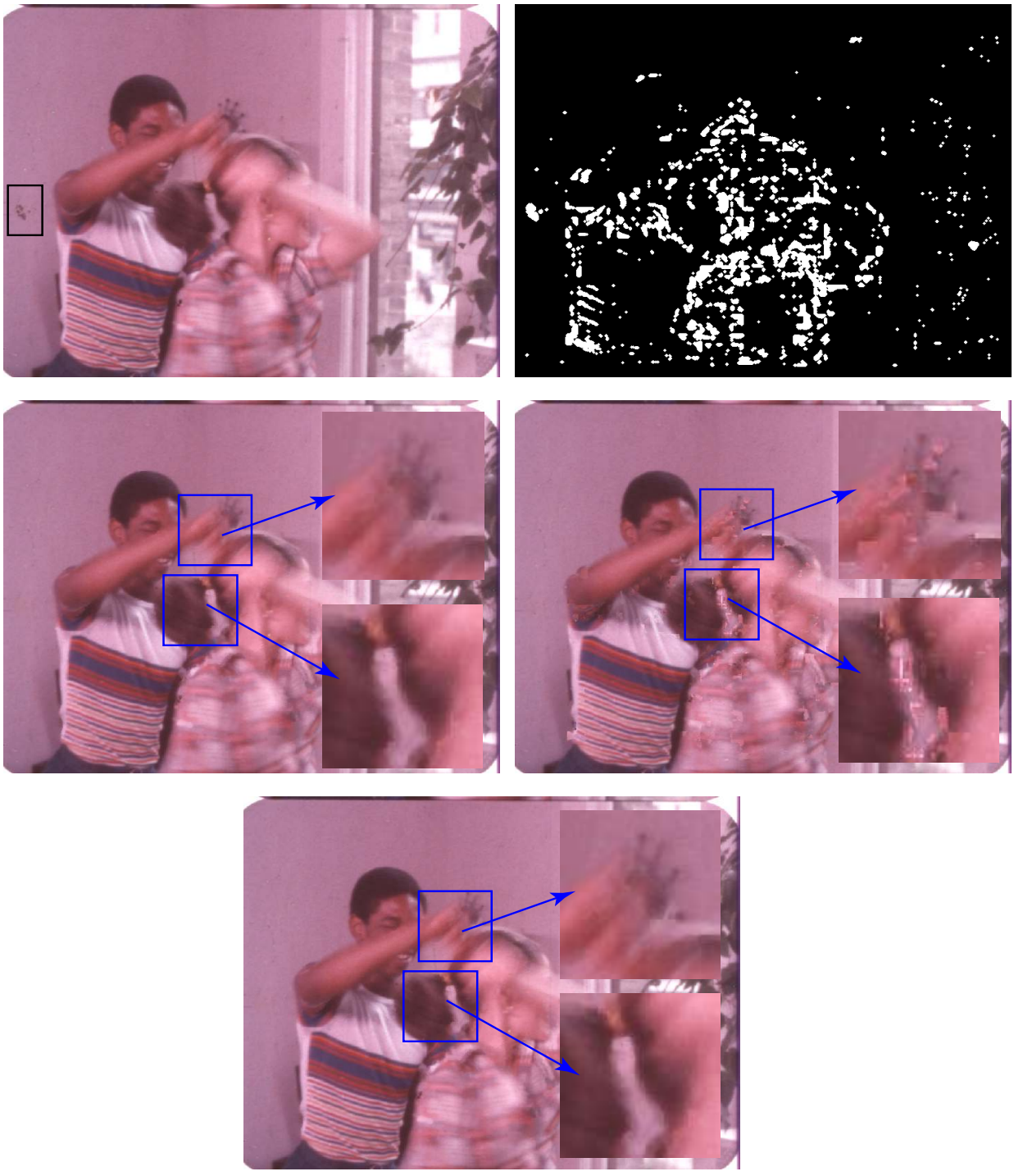


Figure 5.23: *In clockwise direction: Frame 76 of DanceB1 with blotches (shown in black boxes); SDIp detection; BTBR-T restoration; BTBR-F restoration; BTBR-S restoration. Regions shown in blue undergo fast motion and hence generate restoration artifacts in the BTBR-T reconstruction. BTBR-F however favors the spatial restoration for these regions.*

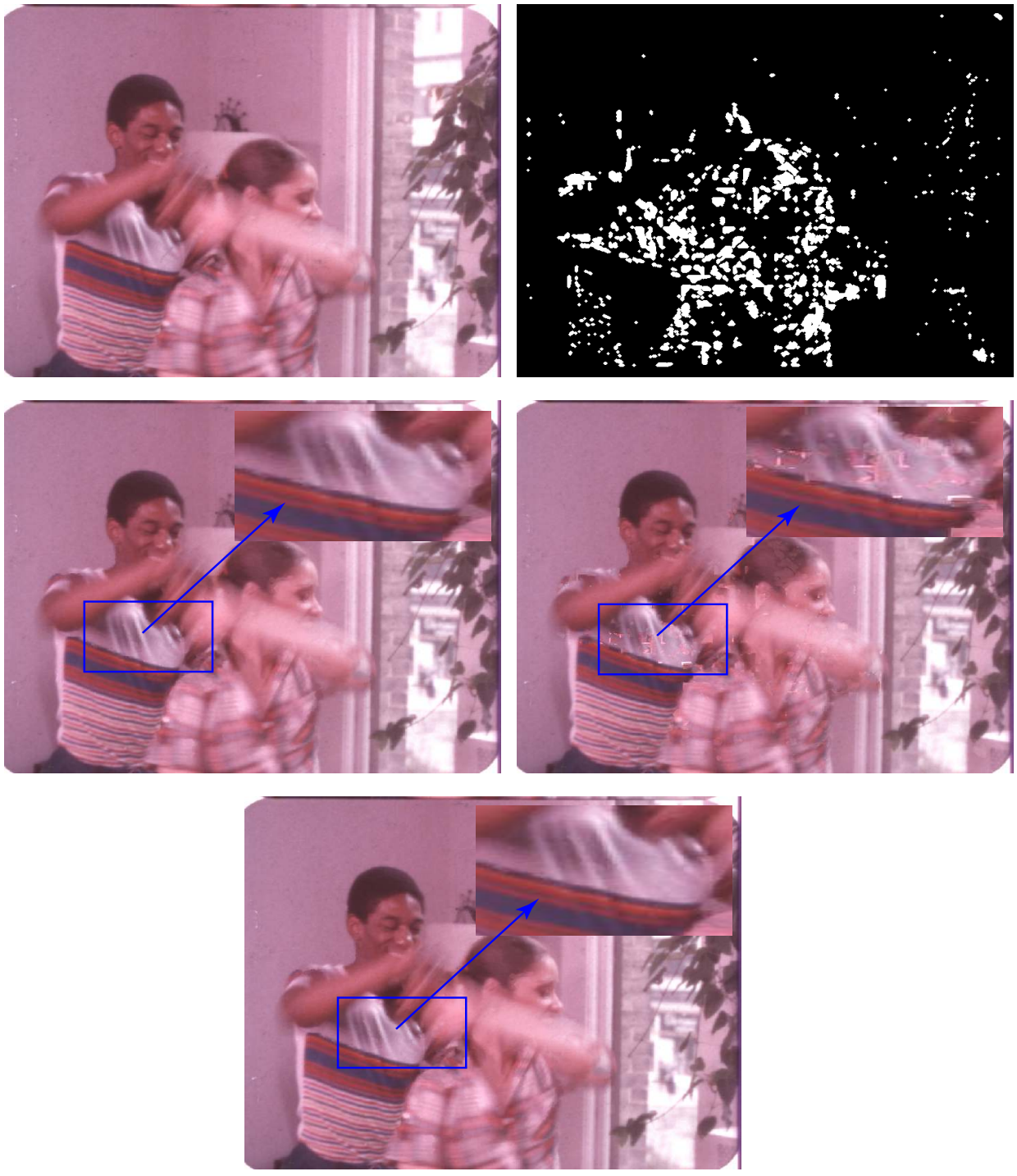


Figure 5.24: *In clockwise direction: Frame 77 of DanceB1; SDIp detection; BTBR-T restoration; BTBR-F restoration; BTBR-S restoration. Regions shown in blue undergo fast motion and hence generate restoration artifacts in the BTBR-T reconstruction. BTBR-F however favors the spatial restoration for these regions.*

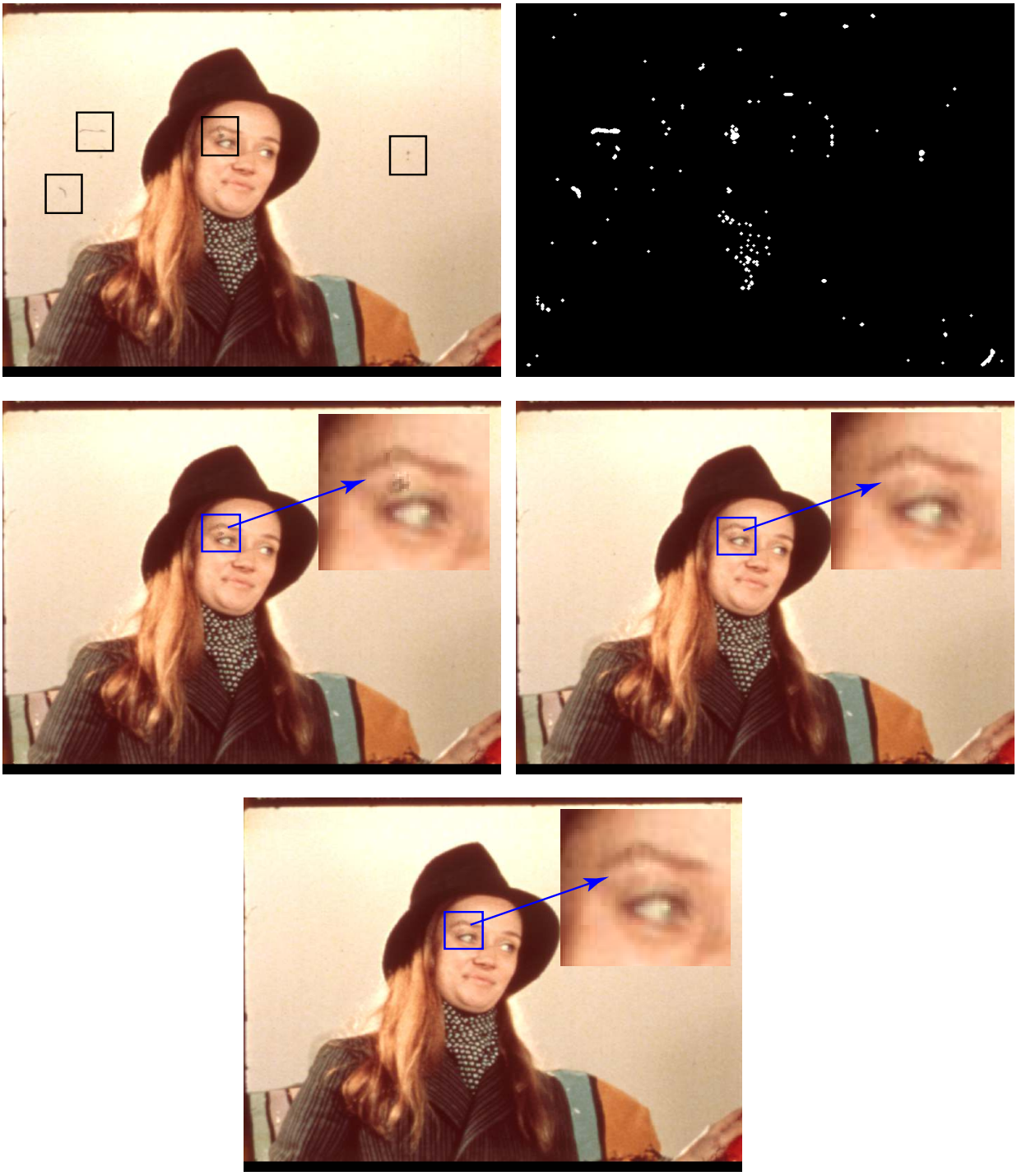


Figure 5.25: *In clockwise direction: Frame 76 of LadyDollB1 with blotches (shown in black boxes); SDIp detection; BTBR-T restoration; BTBR-F restoration; BTBR-S restoration. Spatial removal of BTBR-S failed to fully remove the dark blotch (see blue box). However, BTBR-T removes it successfully using clean information from nearby frames. BTBR-F here favors the BTBR-T reconstruction.*

5.4 Detection Comparison

The estimation of corruption opacity can be regarded as a detection refinement step. Figure 5.26-5.27 show the estimated mattes for some blotches and their corresponding reconstruction using BTBR. The calculated mattes (fourth row) reassemble IR scans of corruptions (third row). This is more evident by examining the regions shown in blue. IR scans here are projected on the opacity corruption domain as previously discussed in section. 5.2. This is done for illustration clarity. As shown in figure 5.26-5.27 (fourth row) the extracted mattes often estimate corruption borders correctly and hence they reject false SDIp detections from the correction step. Note that in figure 5.26 (first column) BTBR misclassifies the region shown in red as being clean. This region lies on a dark clean background and hence BTBR failed to distinguish the dark corruption from the dark original data. The result here is that BTBR did not fully remove this blotch. However the visual impact in the final reconstruction is quiet acceptable as the blotch is consistent in color with the surrounding dark clean regions (see figure 5.26, last row, shown in red).

Figure 5.28 shows the ROC of LabB1 and DanceB1 as generated by SDIp and BTBR. IR sites with less than 20 grayscale values are regarded as ground-truth corrupted locations. A false blotch detection rate of 0.1 is often regarded as a large false alarm rate. As shown in both ROCs BTBR does not reach such high false detection rate. In LabB1 BTBR was able to reduce the false rate of SDIp from 0.16 to around 0.05 while maintaining the correct detection rate of 0.86 (see the last black circled marker in both SDIp and BTBR plots). That is more than 68% reduction in the SDIp false detection rate. In DanceB1 BTBR was able to reduce the false rate of SDIp from 0.16 to around 0.04 while maintaining the correct detection rate of 0.94 (see the last black circled marker in both SDIp and BTBR plots). That is around 75% reduction in the SDIp false detection rate.

Figure 5.29-5.30 shows how BTBR refines SDIp detection masks. Each figure shows two consecutive frames from DanceB1 with blotches shown in blue. In both figures the girl is moving fast and hence she generates a large SDIp false detection rate. BTBR rejects most of this false detection (shown in green, middle row) and correctly detects blotches (shown in white, middle row). Some false detections (shown in red, middle row) are passed to the correction stage. However BTBR correctly estimates the original data in the sites of those false detections (see last row).

5.5 Performance on Grayscale Data

Many real corrupted sequences are old footage and hence they are usually not colored. This section examines how BTBR perform on grayscale data. We generated LabB1G and DanceB1G by projecting LabB1 and DanceB1 on the grayscale domain. Regions that look different in color look less different in grayscale. This often generates more restoration artifacts in grayscale than

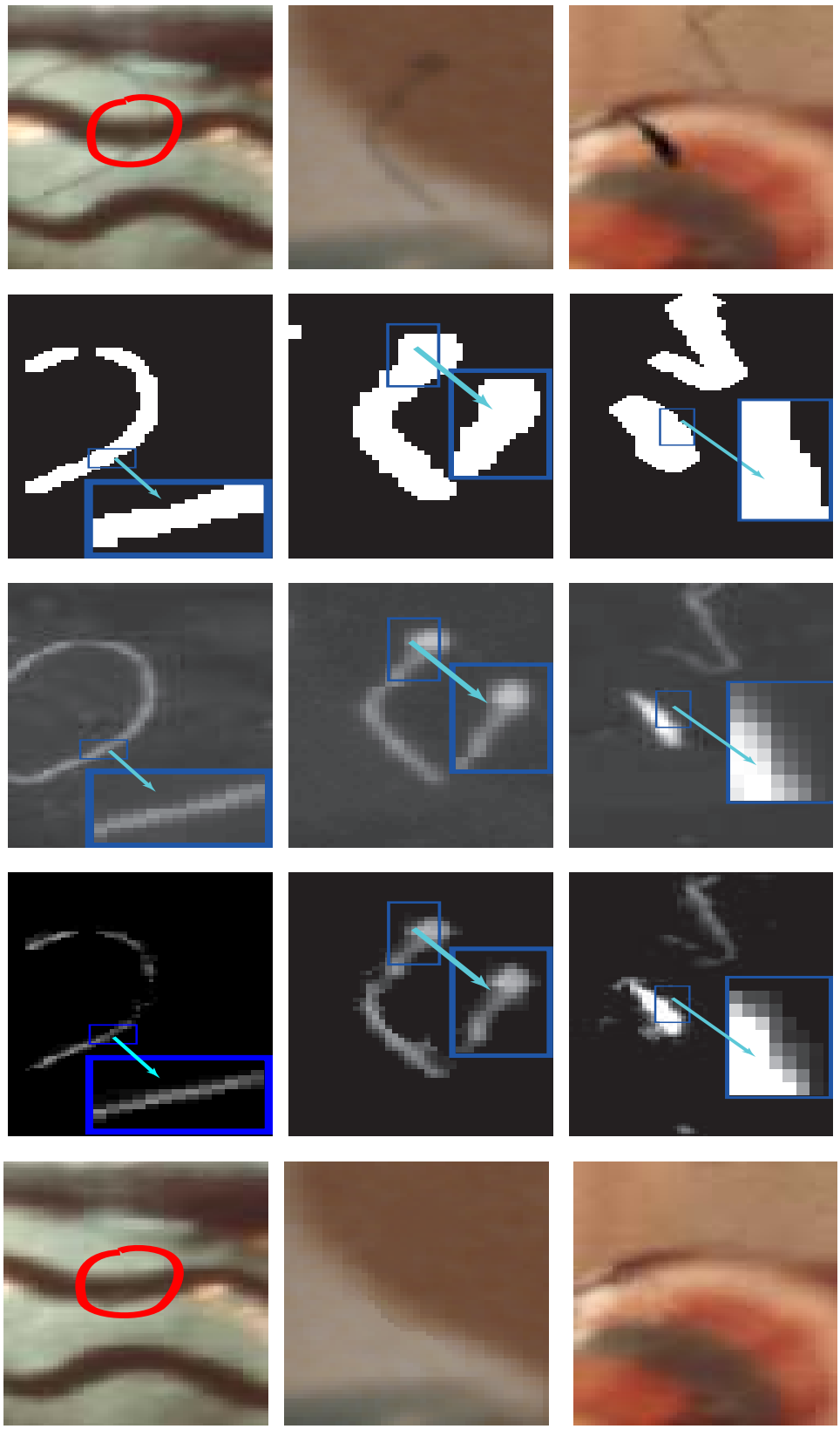


Figure 5.26: *From top: Corrupted data, SDIp masks, IR scans, Corrupted mattes as estimated by BTBR, Corresponding reconstruction. Estimated mattes reassemble the IR scans as shown in blue. BTBR does not remove a dark blotch that is surrounded by dark clean regions (shown in red). This however does not affect the final reconstruction (last row, shown in red).*

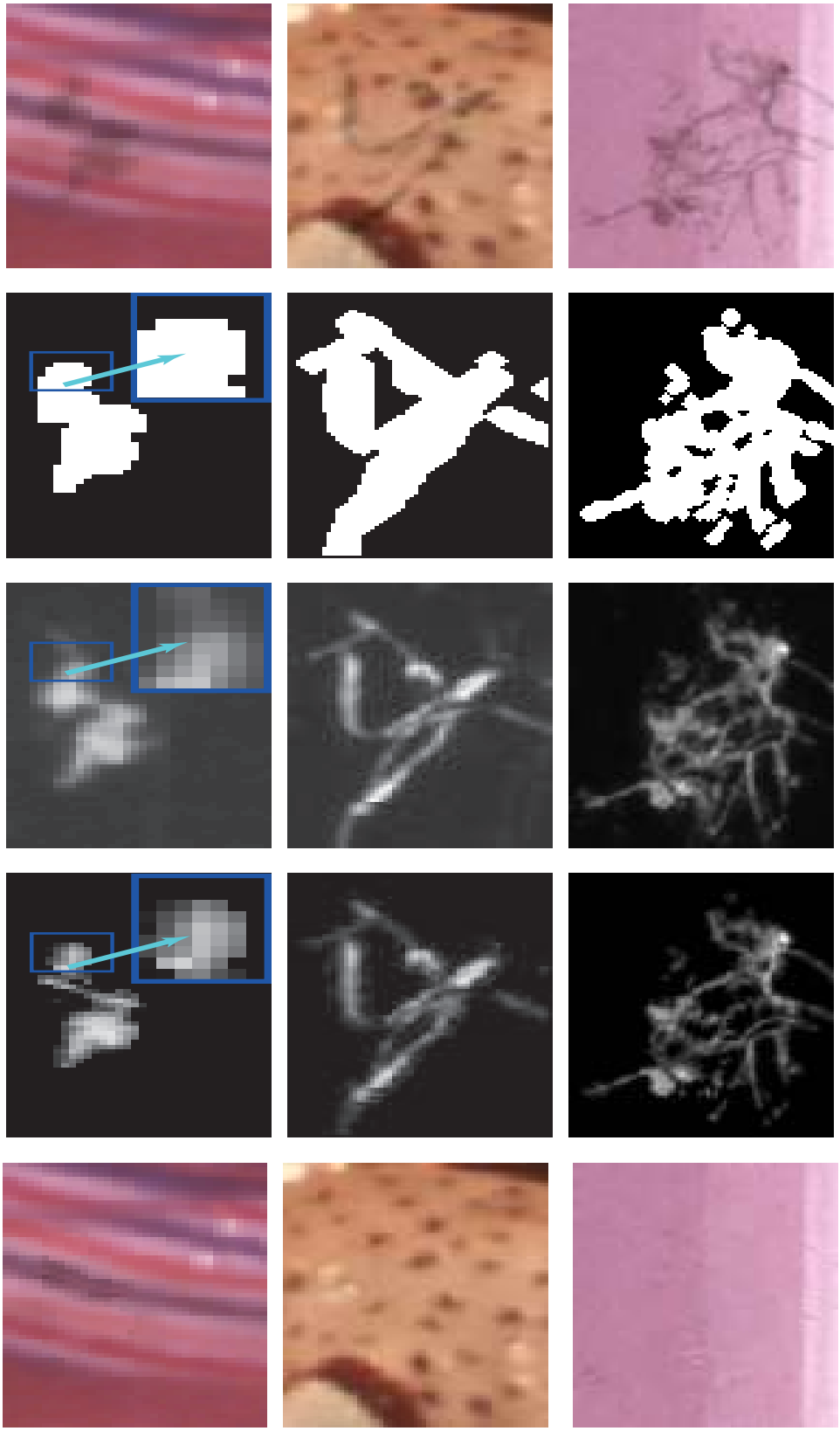


Figure 5.27: *From top: Corrupted data, SDIp masks, IR scans, Corrupted mattes as estimated by BTBR, Corresponding reconstruction. Estimated mattes reassemble the IR scans as shown in blue.*

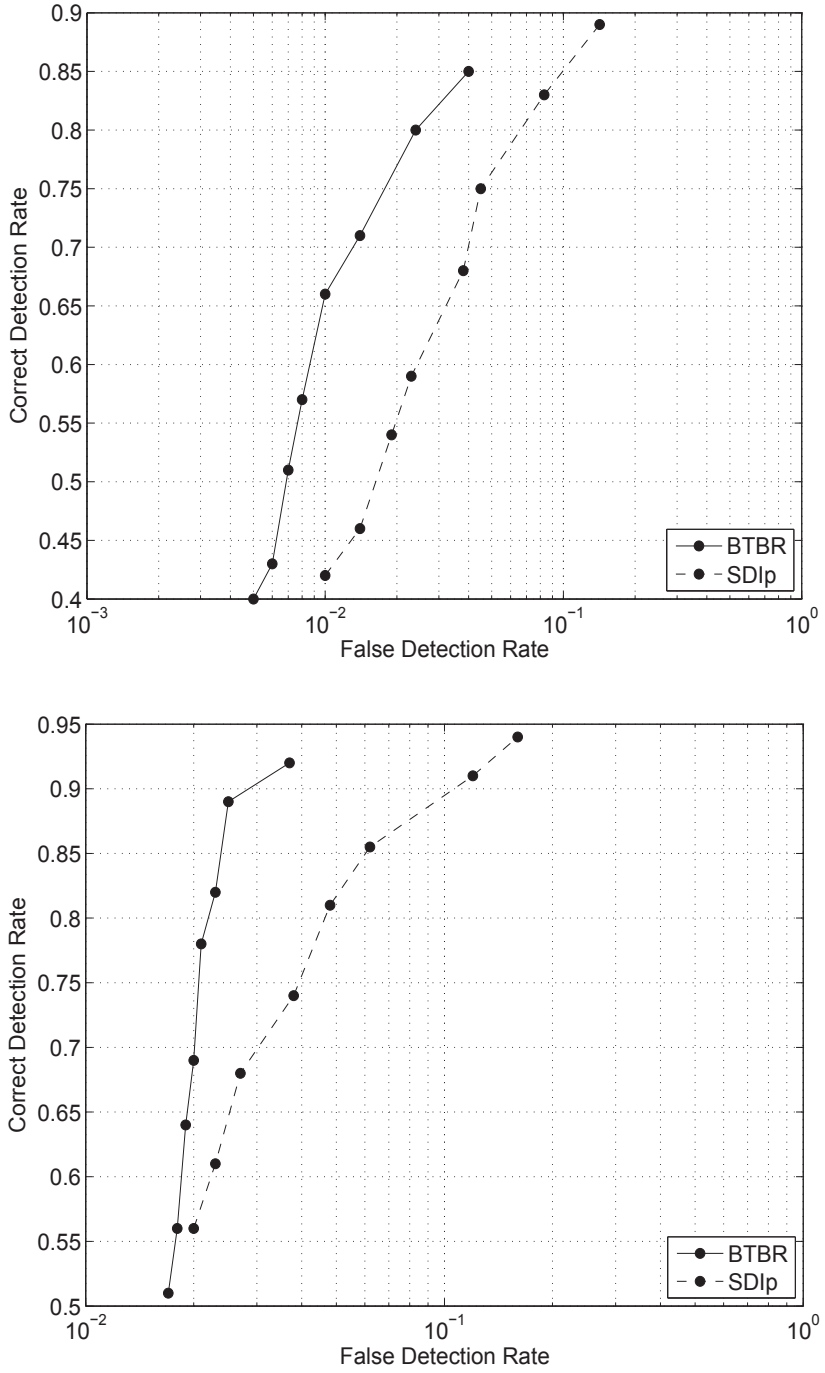


Figure 5.28: *ROC of BTBR and SDIp on LabB1 (Top) and DanceB1 (Bottom). The black circled markers denote the different SDIp thresholds used in ROC evaluation. Here we use 8 SDIp thresholds being [5:2.5:22.5]. Its clear that the refined detections of BTBR have fewer false alarms than SDIp.*

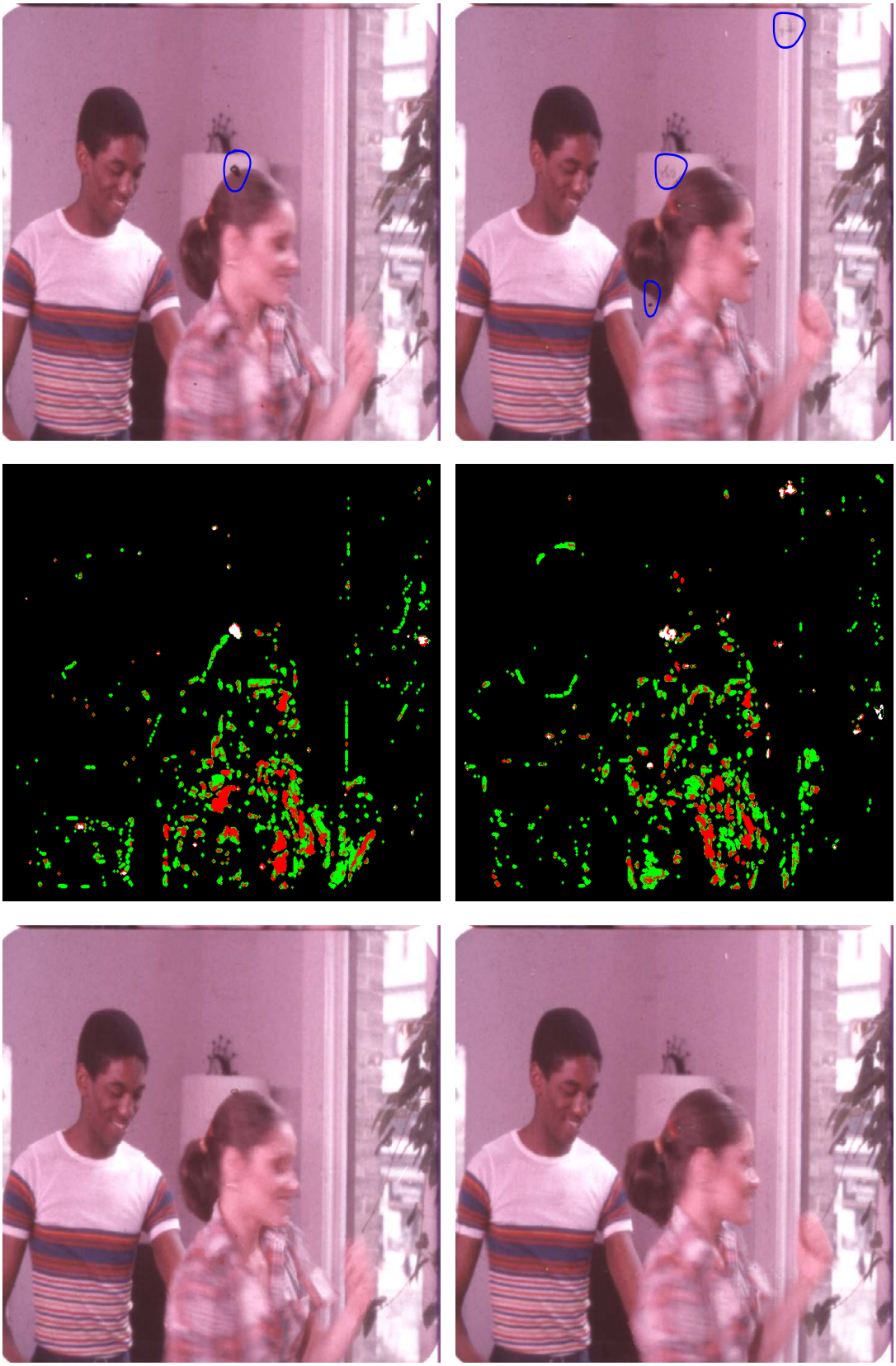


Figure 5.29: *From top: Frame 94 (left) and 95 (right) of DanceB1 with blotches shown in blue; Rejected false SDIp detections (by BTBR) shown in green, correct blotch detections (shown in white) and false BTBR detections (shown in red); Final BTBR reconstruction. BTBR removes blotches and maintains the original data in regions incorrectly classified as corrupted.*

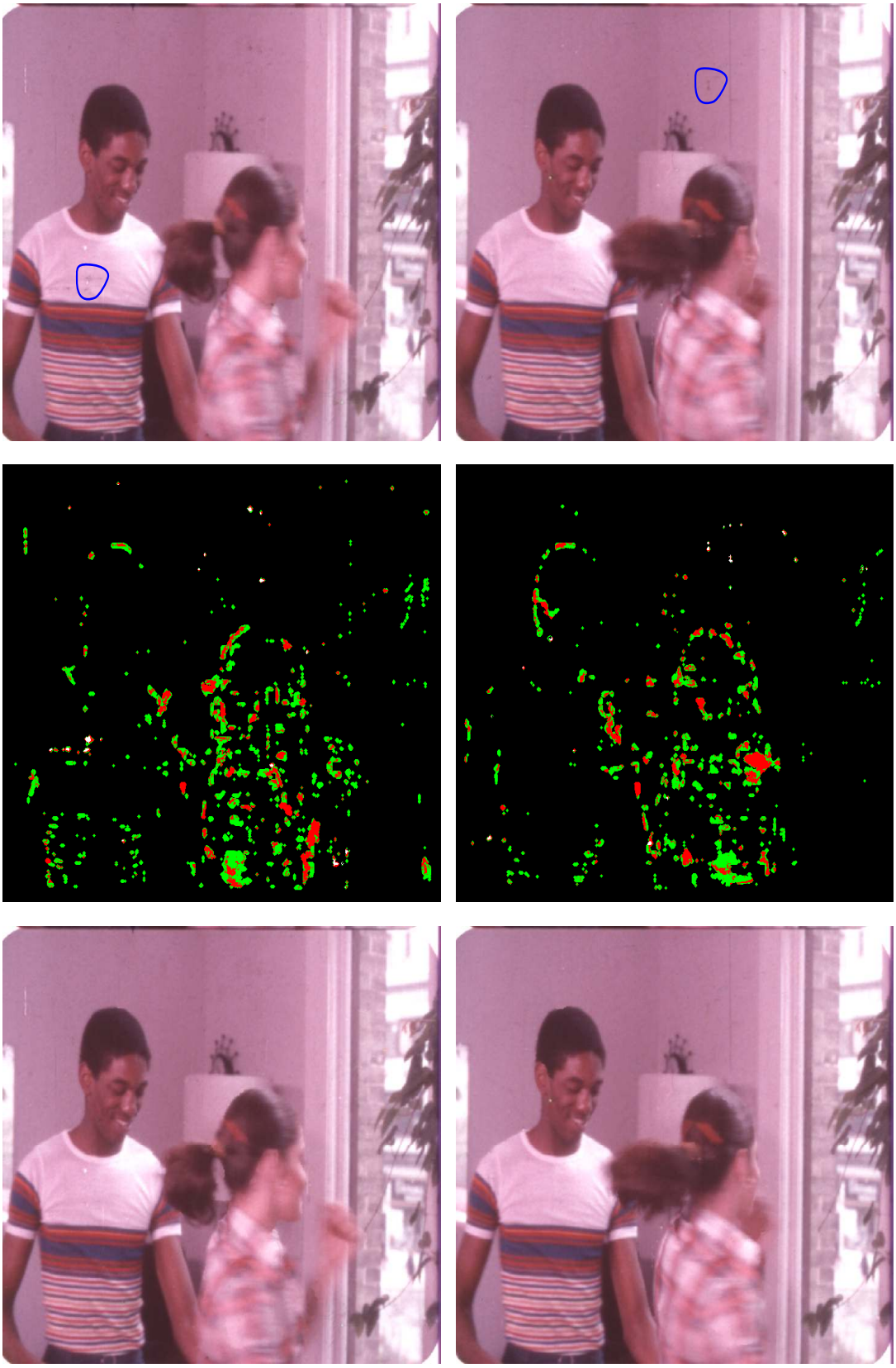


Figure 5.30: *From top: Frame 96 (left) and 97 (right) of DanceB1 with blotch shown in blue; Rejected false SDIp detections (by BTBR) shown in green, correct blotch detections (shown in white) and false BTBR detections (shown in red); Final BTBR reconstruction. BTBR removes blotches and maintains the original data in regions incorrectly classified as corrupted.*

in color as grayscale provides less information than color.

Figure 5.31 shows the original data reconstruction quality of LabB1G and DanceB1G as generated by BTBR. Here we compare the grayscale reconstruction (shown in blue) with the color reconstruction (shown in red). As shown BTBR grayscale reconstruction is a bit worse than the color reconstruction. Figure 5.32 shows reconstruction examples of LabB1G and LabB1. BTBR successfully removed blotches (shown in green) from both grayscale and color sequences (middle and last row respectively). However BTBR generated some noticeable artifacts in LabB1G (shown in red). Here the amount of spatial smoothness imposed on the generated reconstruction was too strong to the point it caused the surrounding white/gray region to bleed into the black clean region. This problem also exists in some cases of color reconstructions (see figure 5.33 left column, shown in blue). However, lowering the level of reconstruction smoothness could lead to incomplete blotch removal as shown in figure 5.33 (right column, shown in green).

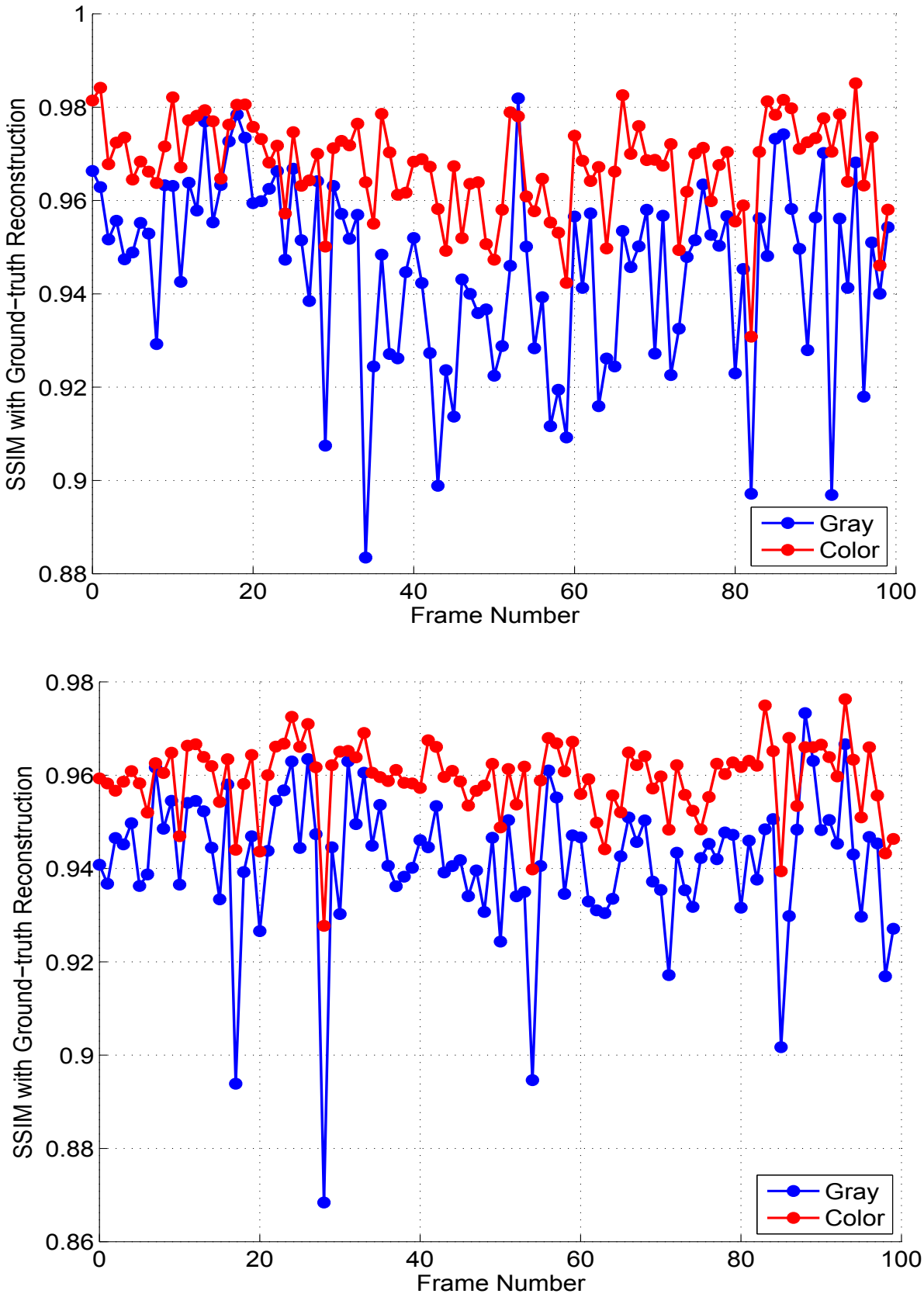


Figure 5.31: *Reconstruction quality of LabB1 (top) and DanceB1 (bottom) as generated by BTBR for both color and grayscale representation (shown in red and blue respectively). As shown color reconstruction is usually better than grayscale reconstruction.*

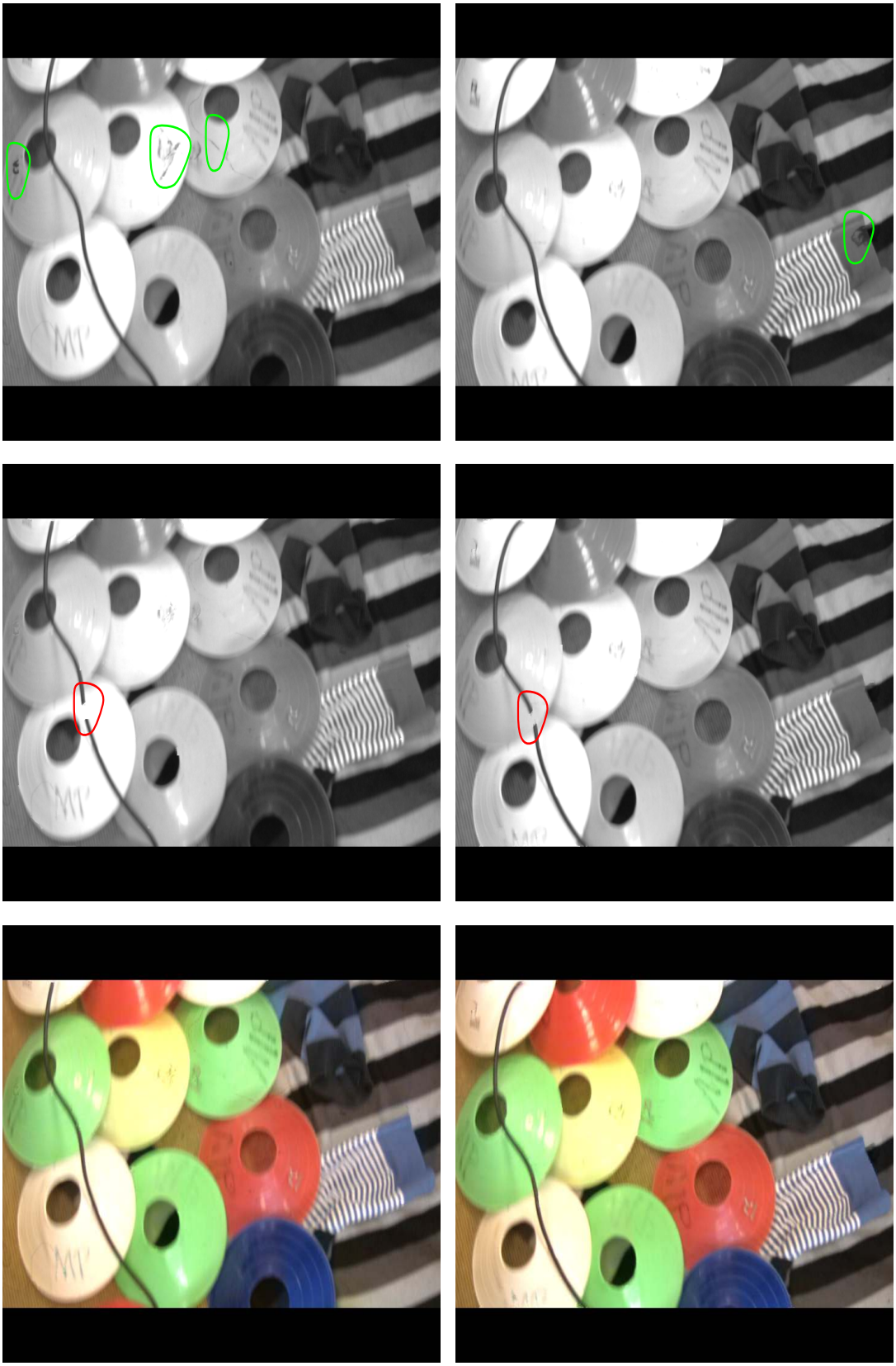


Figure 5.32: *From top; Frame 65 (left column) and 67 (right column) of LabB1G; Grayscale reconstruction using BTBR; Color reconstruction using BTBR. Both grayscale and color reconstructions remove blotches (shown in green). However BTBR generates some restoration artifacts in LabB1G as shown in red.*

Figure 5.33: *Examples of BTBR failure on color sequences. From top: Corrupted frames with blotches (shown in black boxes); BTBR reconstructions. Strong reconstruction smoothness could cause regions to bleed into their neighbors. In the region shown in blue (see left column) the pink region bled into the region of corruption. This problem can be solved by lowering the smoothness level. However, weak smoothness may generate incomplete corruption removal as shown in green (right column).*

5.6 Computational Complexity

Estimation of the set of solution candidates $[\alpha, B]$ for each pel is the most computationally intensive aspect of BTBR, amounting to about 90% of the execution time. The estimation of these candidates is an iterative process which terminates when convergence is reached and so consumes a lot of time. This process is repeated C times where C is the number of solution candidates per pel. Therefore, for a corruption mask of P pels, the number of operations required to generate the solution candidates is $P \times C \times Q$ where Q is average number of iterations to generate one solution candidate. The second most computationally intensive part of the

algorithm is the use of QPBO Graph Cut for optimizing the MAP function.

The average time for processing one colored standard definition frame with BTBR on a 2.33 GHz Quad Core Processor and with an unoptimized MATLAB code is 25.4 seconds. This is taken as the average of processing 50 frames with SDIp masks of threshold 7.5 and with 5 solution candidates (4 spatial and 1 temporal). QPBO is implemented efficiently by using C++ and by processing only the regions of SDIp masks. The average time for processing one colored standard definition frame with JONDI written in an optimized C++ code is around 10 seconds. Hence BTBR is nearly as fast as JONDI. Further reduction in computation for BTCR is possible by reducing the size of corruption mask and reducing the number of solution candidates. By reducing the number of solution candidates from 5 to 3, the the average time for processing one frame dropped from 24.5 to 16.7 seconds. This however has a direct impact on the reconstruction quality especially in textured regions.

5.7 Conclusion

This chapter showed different experiments used in evaluating our corruption removal techniques (BTBR and BTLR). We evaluated our removal techniques against ground-truth estimates of the original data. We presented a technique for generating such ground truth estimates. A relation between dirt opacity and its IR scan is derived. This relation is then used to estimate the amount of corruption/dirt represented by IR scans through the means of a non-binary opacity matte. Results showed that the original underlying data can be estimated by inverting the effect of this matte. The estimated data are a near ground-truth estimate of the original data.

We compared the performance of our corruption removal techniques against four blotch removal techniques (Furnace, BTBR-AOFF, JONDI and BBM) and four line removal techniques (Furnace, BTBR-AOFF, JOMEI and BBM). Reconstruction quality was evaluated against the ground-truth estimates generated from IR scans. Results showed that our techniques generate better reconstruction over all the examined blotch and line removal techniques. This confirms that our dark corruption model used is valid enough. We examined the two main stages of Crawford et al. blotch removal technique [20]. The first stage removes the chroma corrupted component using texture synthesis while the second restores the grayscale channel using a semi-transparent corruption model. We showed that texture synthesis (approximated by BTBR-AOFF) could generate large restoration artifacts in clean regions. We also showed that a semi-transparent grayscale corruption model generates poorer results than a semi-transparent color model. Results show that our algorithms can remove the extremities of blotches very well, in comparison to existing techniques. This brings more robustness to the pathological motion problem.

A limiting factor of our technique is the inability to estimate the exact required amount of background smoothness for optimal reconstruction. This sometimes has an impact on the reconstruction quality as too much smoothness could cause neighboring regions to interfere with

each other while too little smoothness could lead to incomplete removal. It is clear that we could incorporate our model directly into the detection/reconstruction problem along the lines of JONDI [48]. That would imply estimation of texture parameters alongside detection and motion information. Although this might seem a daunting task it provides much potential for future work.

6

Reflection Detection in Image Sequences

This chapter presents a technique for detecting reflections in image sequences. Reflections are often the result of superimposing different semi-transparent layers over each other. As a result the captured image is a mixture between the reflecting surface (background layer) and the reflected image (foreground) (see figure 6.1).

Detecting reflections can be done by looking for characteristics that are specific to reflections. However as reflections can arise by mixing any two images, they come in many shapes and colors (figure 6.1). This makes it difficult to define characteristics that are specific to reflections. In addition, one should be careful when using motion information from reflections. Most motion estimators assume the presence of one motion per pel but in regions of reflections there are two layers moving differently. Hence traditional motion estimators fail in these regions. For these reasons the problem of reflection detection is hard and was not examined before in an explicit form.

Reflections are mixtures of images and therefore they often have low image contrast and poor temporal match of feature points. In addition reflections cannot be described by a physical model as they come in a large variety of shapes and colors. However, they can be described by a number of physical characteristics. In this chapter we present a technique for detecting reflections based on analyzing the spatio-temporal profiles of KLT (Kanade-Lucas-Tomasi) feature points trajectories [56, 85]. We propose that one can detect reflections by examining three main characteristics. The first characteristic is that reflections can be decomposed into two different layers and the second is that they have low image sharpness/contrast. The final characteristic is that regions of reflections often have poor temporal match of feature points.

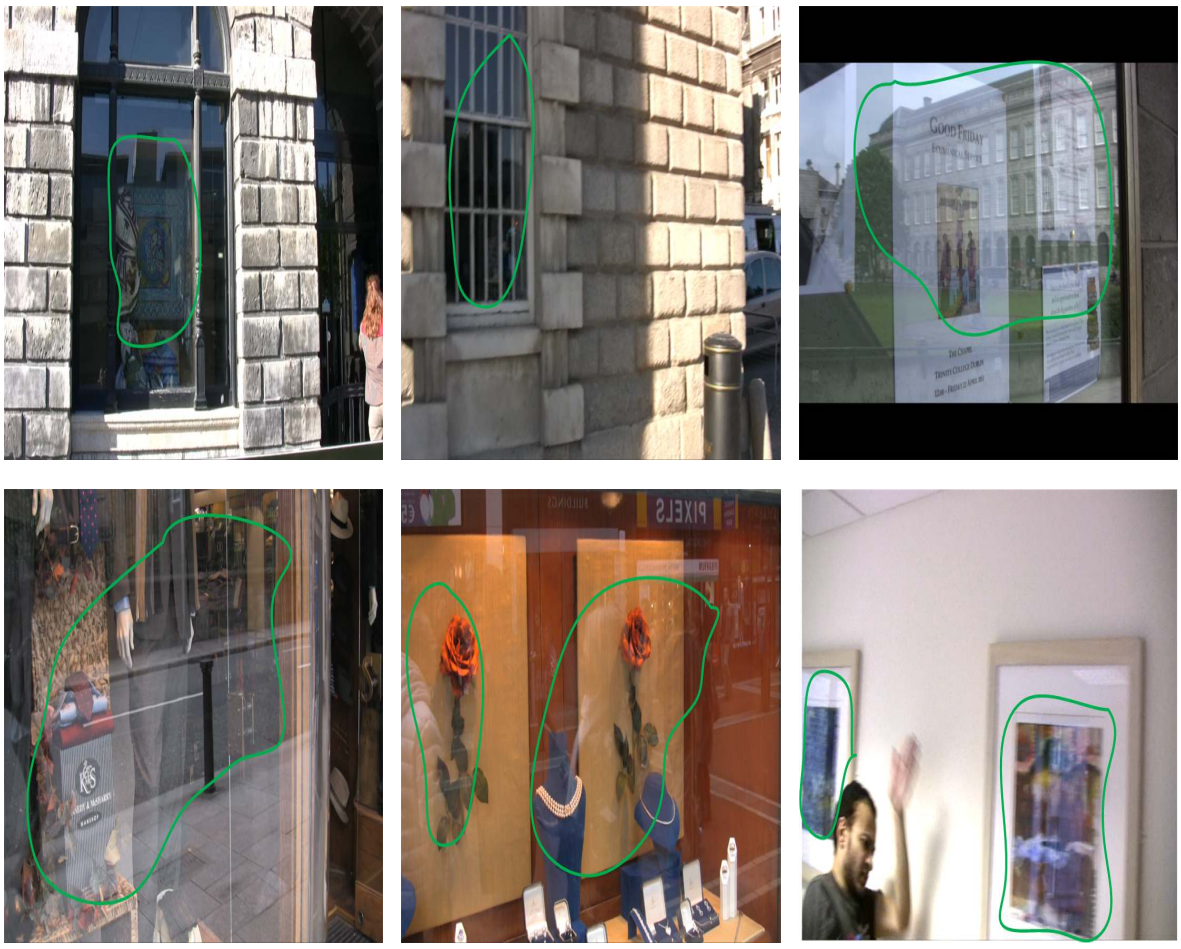


Figure 6.1: *Examples of reflections (shown in green). Reflections come in different forms and colors. This makes their detection a hard problem.*

For layer separation analysis we present and use a separation technique that can operate on a single still image. Most current layer separation techniques either require two mixtures of the same layers under two different mixing conditions [12, 16, 22, 26, 73] or assume a repetitive background motion [74, 83, 94]. These assumptions are often not valid for reflections in real image sequences. However, we noticed empirically that in many real reflections at least one of the layers is more dominant than the other layer in either the red or the blue channel (see figure 6.2 for examples). On the other hand, the green channel often contains a strong blend between the underlying layers as this channel usually contains most of the energy of an image. Hence the source foreground and background layers (F and B) can be estimated by exchanging information between the red and blue channels until the mutual independence between both channels is maximized. Based on this we propose to model the underlying layers as a mixture between the red and blue channels of the observed mixture. We then use the Sarel et al. [73] Layer Information Exchange approach to estimate the underlying layers. We do not claim that

Figure 6.2: *From top: Color image and the red and blue channels respectively. Left; The woman (shown in blue) is the background layer while the reflection on the car window is the foreground layer. Here the foreground layer is more dominant than the background in the blue channel. Right: A foreground layer containing people walking on the street (shown in blue) reflected on a (background) shop indoor. Here the background layer is more dominant than the foreground in the red channel.*

our reflection separation technique could be used to fully remove reflections from videos or to compete against existing separation techniques. What we claim is that the reconstructed layers can be useful for reflection detection since reflection is reduced in those layers.

We propose several weak reflection detectors based on analyzing the three main characteristics of reflections previously discussed. A final strong detector is generated by combining the weak detectors. The problem is formulated within a Machine Learning framework and spatio-temporal information are used in a way to reject false alarms and to generate detections that are consistent in both space and time. We also show how to detect missed reflections via minimal manual intervention. This is important in a user interactive environment e.g. post-production, to correct for areas containing few feature point trajectories. Several sequences are processed and results show high detection rates with rejection of complicated motion patterns e.g. motion blur, occlusion, fast motion.

In the next section we propose our layer separation technique. We then present our reflection detection technique followed by results.

6.1 Layer Separation through Color Independence

The foreground (F) and background (B) layers of an examined image mixture (reflection) M is modeled as a linear combination between the red and blue channels of M as follows

$$\begin{aligned} F &= a_{11}M_r + a_{12}M_b \\ B &= a_{21}M_r + a_{22}M_b \end{aligned} \tag{6.1}$$

Here $[a_{11}, a_{12}, a_{21}, a_{22}]$ are the mixing parameters while M_r and M_b are the red and blue channels of M respectively. We estimate the source layers $[F, B]$ up to a scale by exchanging information between the red and blue channels of M until the mutual independence between both channels is maximized. The estimated layers are not color images, instead they are images containing the structures of the underlying foreground and background. We estimate $[F, B]$ using Sarel et al.'s 'Layer Information Exchange technique'. This technique was discussed in detail in section 2.2.1.1 and is reformulated for our problem as follows

$$\begin{aligned} \hat{F} &= M_r - \gamma_1 M_b \\ \hat{B} &= M_b - \gamma_2 M_r \end{aligned} \tag{6.2}$$

Here $[\hat{F}, \hat{B}]$ are the estimates of $[F, B]$ while $[\gamma_1, \gamma_2]$ are the separation parameters to be estimated. Note that in equation 6.2 we assume that F is more dominant than B in M_r while B is more dominant than F in M_b.

Images are processed in 50×50 blocks. For each block an exhaustive search for $[\gamma_1, \gamma_2]$ is performed. We first solve for the γ_1 that generates the best F estimate and then solve for the γ_2 that generates the best B estimate. Motivated by the work of Levin et al. on layer

separation [54], the best separated layer is selected as the one with the lowest cornerness value. The Harris corner operator is used here. A minimum texture is imposed on the separated layers by discarding layers with a variance less than T_x. For an 8-bit image, T_x is set to 2. The removal of this constraint can generate empty/null meaningless layers.

6.1.1 Results

6600 image mixtures of size 50×50 pels were created by mixing real natural images using the image compositing equation which is re-stated here as follows

$$M = \alpha F + (1 - \alpha)B \tag{6.3}$$

Here M is the mixture generated by $[F, B]$ while α is the mixing parameter. $[F, B]$ are 50×50 image patches randomly selected from a database of natural images. Most natural images used here are of size 576×720 pels. For each layer pair $[F, B]$ we generate 11 different mixtures using $\alpha = [0 : 0.1 : 1]$. We call this database SynRef1. Examples from this database are in Appendix B (see figure 10.1). The artificially created mixtures are processed by our layer separation technique and the reconstruction error ξ for a mixture M is calculated as follows

$$\xi = \min(\xi_F, \xi_B) \tag{6.4}$$

$$\xi_F = \text{MSE}(\hat{F}, a_{11}M_r + a_{12}M_b) \tag{6.5}$$

$$\xi_B = \text{MSE}(\hat{B}, a_{21}M_r + a_{22}M_b) \tag{6.6}$$

Here $[\xi_F, \xi_B]$ is the reconstruction mean squared error (MSE) of $[\hat{F}, \hat{B}]$ respectively. As our technique estimates $[F, B]$ up to a scale, $[a_{11}, a_{12}, a_{21}, a_{22}]$ are estimated in a way such that ξ_F and ξ_B are minimized. This is done using the standard MATLAB function `lsqnonlin`.

Figure 6.3 shows the reconstruction MSE averaged over the 6600 examined synthetic mixtures. For comparison we use the Layer Information Exchange technique using three features being GNGC, edges and corners. We also show the recorded MSE obtained with no separation. No separation implies that the estimated layers are set to the red and blue channels of the examined mixtures. Results show our separation technique reduces reflections/image mixing. In addition, minimizing corners in the separated layers generates better separation than minimizing edges or GNGC.

Figures 6.4-6.5 show separation results generated by the proposed technique on real data. Note that because extracted layers are a combination of the red and blue channels they will never resemble the 'true' visible light layers. However we ask the reader to recall that this layer extraction exercise is only a step that will eventually allow us to generate a model-based type feature to be used in reflection detection. Blocking artifacts are due to processing images in 50×50 blocks. These artifacts are irrelevant to reflection detection. We show either F or B for each examined mixture as we will only use the best separated layer in our reflection detection technique. Figure 6.4-6.5 show that reflections (shown in blue) are reduced in the separated layers.

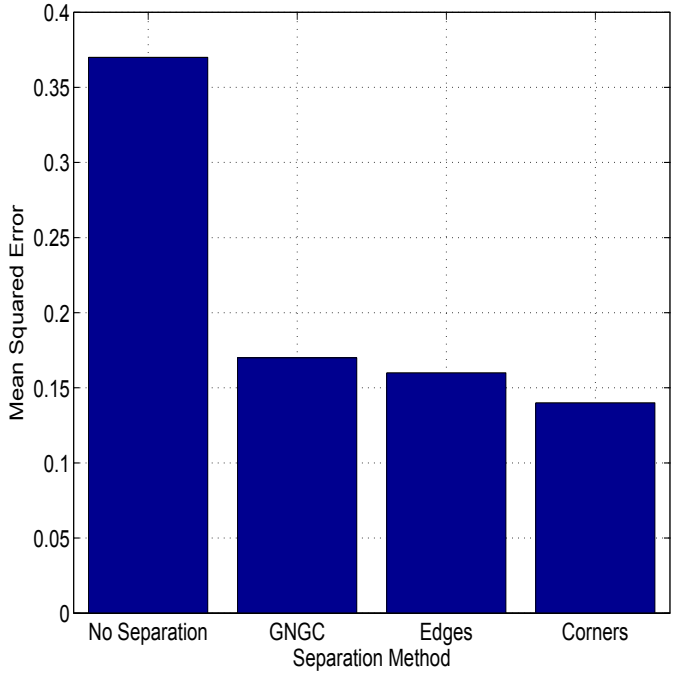


Figure 6.3: *Reconstruction MSE averaged over 6600 synthetic mixtures. The first bar shows MSE with no separation. The other three bars shows separation results by minimizing GNGC, edges and corners in the separated layers. Our technique (Corners) generates the lowest reconstruction error.*

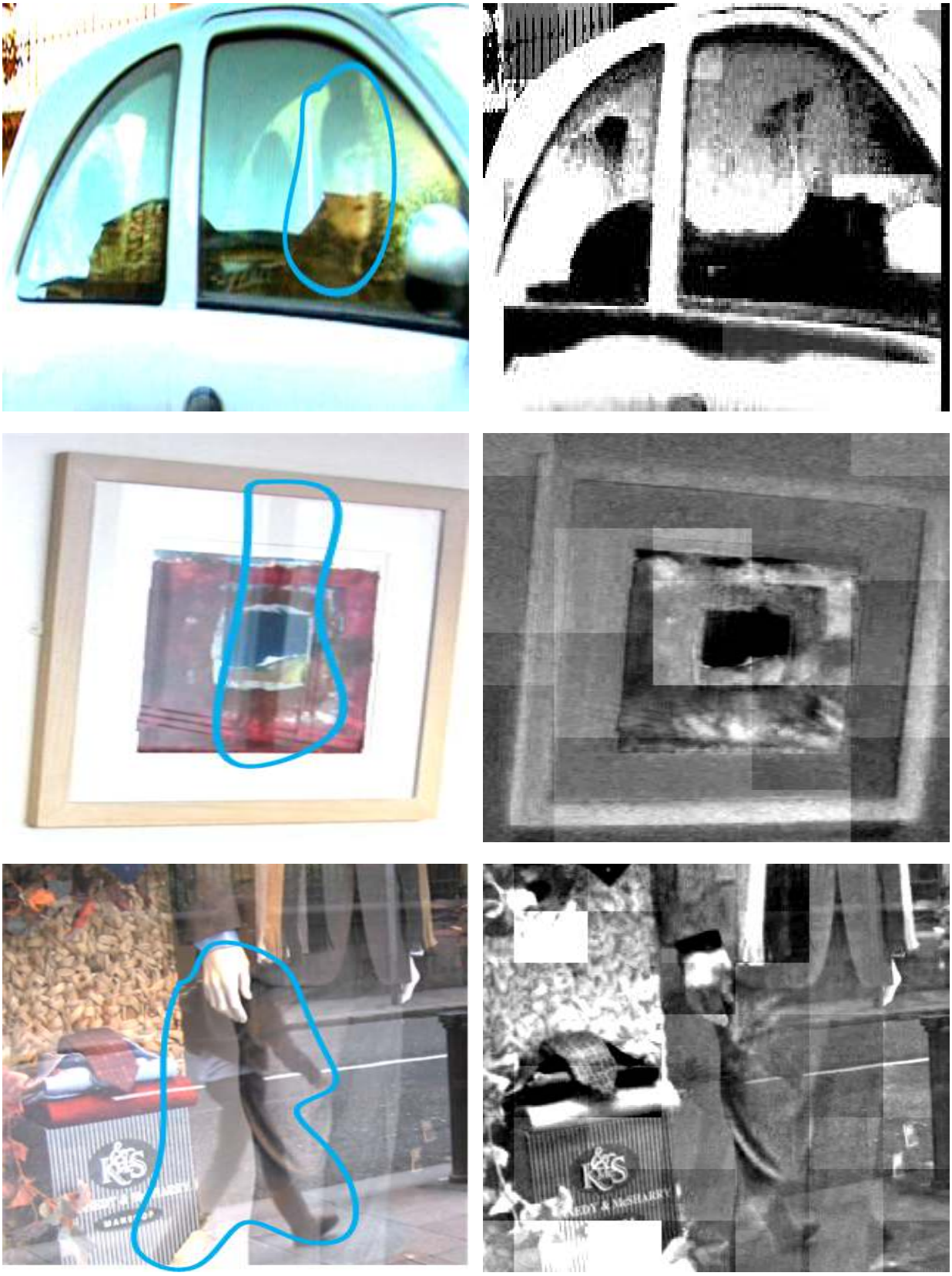


Figure 6.4: *Reducing reflections using the proposed layer separation technique. Color images are the original images with reflections (shown in blue). The uncolored images represent one source layer (calculated by our technique) with reflections reduced. Blocking artifacts are due to processing images in blocks of* 50×50.

Figure 6.5: *Reducing reflections using the proposed layer separation technique. Color images are the original images with reflections (shown in blue). The uncolored images represent one source layer (calculated by our technique) with reflections reduced. Blocking artifacts are due to processing images in blocks of* 50×50.

6.2 Inference for Reflection Detection

The goal of the algorithm is to find regions in image sequences that contain reflections. This is achieved by examining trajectories of feature points. Trajectories are generated using the KLT feature point tracker [56, 85]. We use KLT trajectories as they are fast to calculate. Denote $\mathcal{P}_n^i$ as the feature point of the i^{th} track in frame n and Ω_n^i as the 50×50 image patch centered on $\mathcal{P}_n^i$. For each $\mathcal{P}_n^i$, analyses are carried out over a frame window. Based on the outcome, a binary label field l_n^i is assigned to each patch Ω_n^i. l_n^i is set to 1 for reflection and 0 otherwise.

The problem is formulated in a Machine Learning framework. We estimate l based on information extracted from Ω using several weak detectors. Those detectors are combined using an Adaboost framework to yield a strong detector. The final decision rule at each patch is derived from a consensus among three detectors, one of which is the Adaboost output. In a Bayesian mindset we would have combined measurements at each patch Ω with our prior knowledge of l. With that in mind we take the output of the Machine Learning process and incorporate spatial and temporal smoothness in post-processing steps.

We now describe each stage of our technique in detail. In the next section we discuss the weak detectors $D_1 - D_9$. We then go to describe our combined detector. We show how to improve detection by imposing various spatial and temporal smoothness constraints on the generated labels l. Finally, we show how to detect missed detections by very slight manual intervention.

6.2.1 Feature Point Analyses for Reflection Detection

We detect reflections by examining three main characteristics. The first characteristic is that reflections can be decomposed into two independent layers and the second is that they have low image sharpness/contrast. The final characteristic is that regions of reflections often have poor temporal match of feature points. In this section we present 9 weak reflection detectors, each detector measures one of the three main reflection characteristics. Analyses are performed on every point along every KLT trajectory. To avoid classifying pathological motion as being reflection, all analyses are performed on feature point trajectories of length more than 4 frames.

We use the structural similarity measure SSIM [93] in some of our weak detectors. The SSIM between two examined images $[I_1, I_2]$ is denoted by $\text{SSIM}(I_1, I_2)$. SSIM returns a value between $0 - 1$ where 1 denotes identical similarity between the examined images. In this chapter SSIM is often used to assess the quality of separated layers. However as all layer separation techniques estimate layers up to a scale, we only compare the structures of the examined images. This is done by turning off the luminance component of SSIM wherever it is used in this section.

We evaluate the performance of each weak detector by processing a database of synthetically created reflections. We call this database SynRef2. SynRef2 consists of four image sequences each containing 50 frames of size 576×720 pels. For each sequence regions of reflections are created by mixing two different reflection-free sequences using the compositing equation (see equation 6.3). We set the blending factor α to 0.4 in the four artificially created sequences.

SynRef2 contains 89393 feature point patches. 35966 out of those patches contain artificially created reflections. Frames from SynRef2 are in Appendix B (figure 10.2). We now describe each weak detector in detail.

6.2.1.1 Layer Separation through Color Independence (Detector D_1)

Our separation technique (presented in section 6.1) is used to decompose the examined image patch Ω_n^i into two (foreground and background) layers $\mathcal{F}_n^i$ and $\mathcal{B}_n^i$. Patches containing reflection are defined as ones with lower temporal discontinuity after separation than before separation. The difference in temporal discontinuity in the examined layer $\mathcal{L}$ (either foreground or background) before and after separation is denoted by d_1 and is defined as follows

$$d_1 = \max(\text{SSIM}(\mathcal{G}_n^i, \mathcal{G}_{n-1}^i), \text{SSIM}(\mathcal{G}_n^i, \mathcal{G}_{n+1}^i)) - \max(\text{SSIM}(\mathcal{L}_n^i, \mathcal{L}_{n-1}^i), \text{SSIM}(\mathcal{L}_n^i, \mathcal{L}_{n+1}^i)) \quad (6.7)$$

where

$$\begin{aligned} \text{SSIM}(\mathcal{L}_n^i, \mathcal{L}_{n-1}^i) &= \max(\text{SSIM}(\mathcal{F}_n^i, \mathcal{F}_{n-1}^i), \text{SSIM}(\mathcal{B}_n^i, \mathcal{B}_{n-1}^i))) \\ \text{SSIM}(\mathcal{L}_n^i, \mathcal{L}_{n+1}^i) &= \max(\text{SSIM}(\mathcal{F}_n^i, \mathcal{F}_{n+1}^i), \text{SSIM}(\mathcal{B}_n^i, \mathcal{B}_{n+1}^i)) \end{aligned} \quad (6.8)$$

Here $\mathcal{G} = 0.1\Omega_r + 0.7\Omega_g + 0.2\Omega_b$ where $[\Omega_r, \Omega_g, \Omega_b]$ are the red, green and blue components of Ω respectively. The detector discussed here is named D_1 and flags a regions as reflection if d_1 is less than a threshold T_1. More explicitly, D_1 is defined as follows

$$D_1 = d_1 < T_1. \quad (6.9)$$

Figure 6.6 shows the ROC of D_1 as generated from processing SynRef2. Here D_1 is evaluated at $T_1 = [-0.2044 : 0.003 : 0]$.

6.2.1.2 Intrinsic Layer Extraction (Detector D_2)

Let Γ_i denote the (static) (either foreground or background) layer extracted by processing the 50×50 i^{th} track using the Yair separation technique [94]. In the case of reflection the structure similarity d_2 between the observed mixture at the examined patch Ω_n^i and Γ_i should be low. d_2 is defined as follows

$$d_2 = \text{SSIM}(\Omega_n^i, \Gamma_i) \quad (6.10)$$

where SSIM is the structural similarity measure [93] with the luminance component turned off. The detector discussed here is named D_2 and flags Ω_n^i as containing reflection if

$$D_2 = d_2 < T_2 \quad (6.11)$$

where T_2 is a threshold. Figure 6.6 shows the ROC of D_2 as generated from processing SynRef2. Here D_2 is evaluated at $T_2 = [0.32 : 0.01 : 1]$.

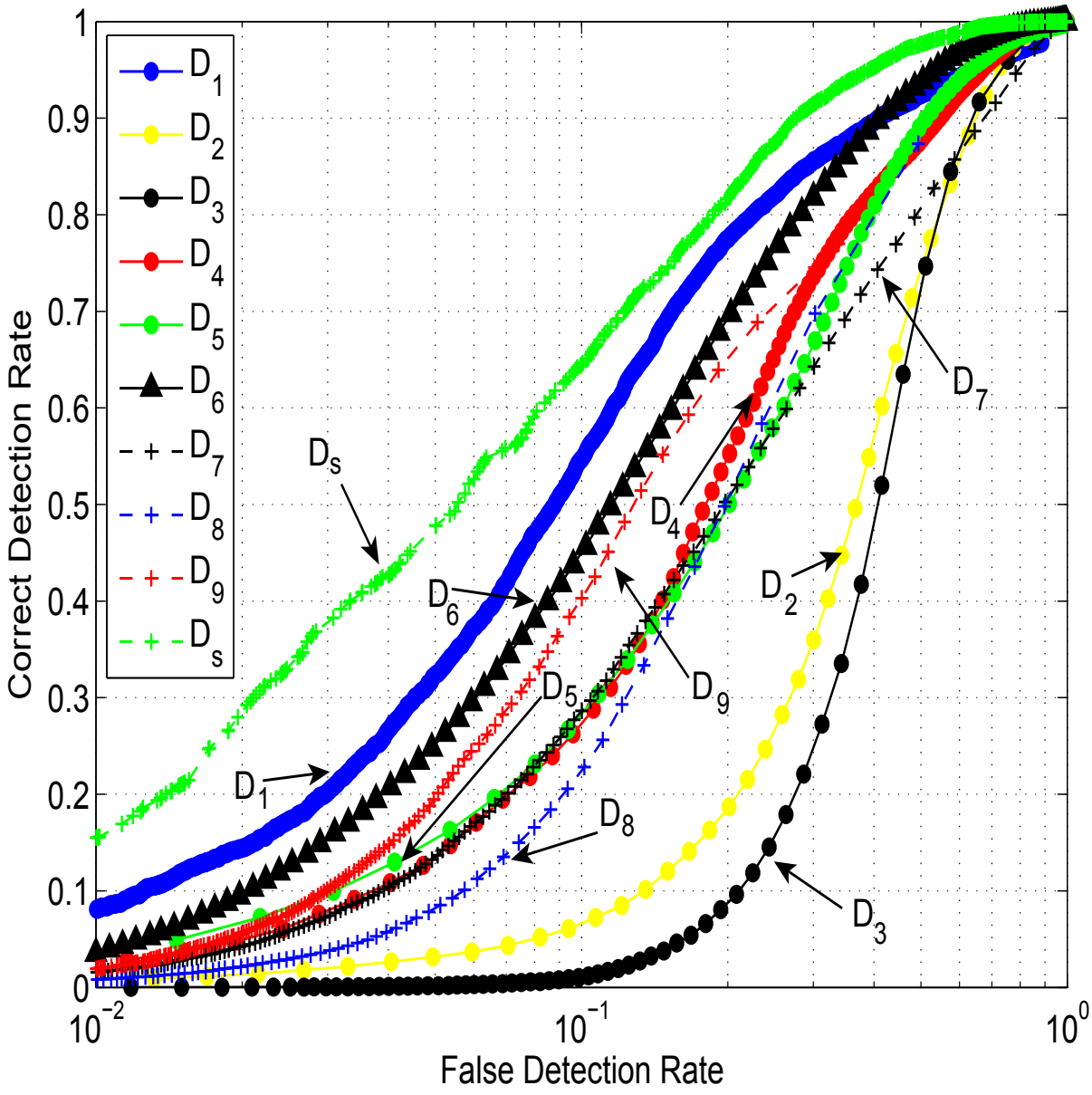


Figure 6.6: *ROC for D_{1-9} and D_s. The Adaboost detector D_s outperforms all other detectors and D_1 is the second best detector.*

6.2.1.3 Color Channel Independence (Detector D_3)

This detector is based on the observation that in regions of reflections one layer is usually more dominant than the other in either the red or the blue channel. Hence this detector flags the examined Ω_n^i as containing reflection if its red and blue channels have low structural correlation. Structural correlation is measured using the Generalized Normalized Cross Correlation (GNGC) of Sarel et al. [73]. GNGC takes values between 0 and 1 where 1 denotes a perfect match between the red and blue channels and hence no reflection. The detector presented here is named D_3

and is expressed as follows (where n and i are dropped for clarity)

$$D_3 = GNGC(\Omega_r, \Omega_b) < T_3 \tag{6.12}$$

Here $[\Omega_r, \Omega_b]$ are the red and blue channels of the examined Ω_n^i and T_3 is a threshold. Figure 6.6 shows the ROC of D_3 as generated from processing SynRef2. Here D_3 is evaluated at $T_3 = [0.38 : 0.01 : 1]$.

6.2.1.4 Image Sharpness (Detectors D_4 and D_5)

As reflections are mixtures of different layers they often have low image contrast/sharpness. Two detectors measuring image sharpness are proposed. The first, D_4, estimates the first order derivatives for the examined patch Ω_n^i and flags it as containing reflection if the mean of the gradient norm within the examined patch is $< T_4$ where T_4 is a threshold. The second detector D_5 estimates image sharpness with the metric of Ferzil et. al. [27] and flags a patch as reflection if its sharpness value is $< T_5$. This sharpness metric [27] is a perceptual-based metric that is appropriate for human decisions. It takes values between 0 (weak sharpness) and 1 (strong sharpness).

Figure 6.6 shows the ROC of D_4 and D_5 as generated from processing SynRef2. Here D_4 and D_5 are evaluated at $T_4 = [2.0 : 0.1 : 36.7]$ and $T_5 = [0.05 : 0.01 : 1]$ respectively.

6.2.1.5 SIFT Temporal Profile (Detector D_6)

In regions of reflections, feature points in successive frames do not match well. The detector presented here (D_6) flags the examined patch Ω_n^i as reflection if its SIFT features [55] exhibit high temporal mismatch. Lowe [55] assigns a vector $\mathbf{p} = [\mathbf{x}, \mathbf{s}, \mathbf{g}]$ to every SIFT point in Ω_n^i. The vector contains the position of the examined point $\mathbf{x}= [x, y]$, scale and dominant orientation from the SIFT descriptor, $\mathbf{s} = [\delta, o]$, and the 128 point SIFT descriptor $\mathbf{g}$. SIFT points are matched with the neighboring frames using [55]. The average distance d_6 between the matched vectors is defined as follows (where n and i are dropped for clarity)

$$d_6 = \frac{1}{K} \sum_{j=1}^{K} \min(\|p_n^j - p_{n-1}^j\|, \|p_n^j - p_{n+1}^j\|) \tag{6.13}$$

Here K is the number of SIFT features in the examined patch, p_{n-1}^j is the SIFT feature in frame $n-1$ matched with the j^{th} SIFT feature in the examined frame. p_{n+1}^j is defined similarly. D_6 flags the examined patch Ω as containing reflection if

$$D_6 = d_6 > T_6 \tag{6.14}$$

where T_6 is a threshold. Figure 6.6 shows the ROC of D_6 as generated from processing SynRef2. Here D_6 is evaluated at $T_6 = [0.0038 : 0.005 : 0.5888]$.

6.2.1.6 Grayscale Temporal Profile (Detector D_7)

Reflections often have poor temporal match between frames. The detector presented here (D_7) flags the examined image patch Ω_n^i as reflection if its grayscale profile does not change smoothly through time. The temporal grayscale change d_7 is defined as follows

$$d_7 = \min(\|\mathcal{C}_n^i - \mathcal{C}_{n-1}^i\|, \|\mathcal{C}_n^i - \mathcal{C}_{n+1}^i\|) \tag{6.15}$$

Here $\mathcal{C}_n^i$ is the mean value for $\mathcal{G}_n^i$, the grayscale representation of Ω_n^i. D_7 flags Ω_n^i as reflection if d_7 is more than a threshold T_7. More explicitly D_7 is defined as

$$D_7 = d_7 > T_7 \tag{6.16}$$

Figure 6.6 shows the ROC of D_7 as generated from processing SynRef2. Here D_7 is evaluated at $T_7 = [0.0952 : 0.1 : 13.1952]$.

6.2.1.7 AutoCorrelation Temporal Change (Detector D_8)

D_8 flags the image patch Ω_n^i as reflection if its grayscale ($\mathcal{G}_n^i$) autocorrelation is undergoing large temporal change. The temporal change in the autocorrelation d_8 is defined as follows

$$d_8 = \sqrt{\min(\frac{1}{N}\|\mathcal{A}_n^i - \mathcal{A}_{n-1}^i\|^2, \frac{1}{N}\|\mathcal{A}_n^i - \mathcal{A}_{n+1}^i\|^2)} \tag{6.17}$$

$\mathcal{A}_n^i$ is a vector containing the autocorrelation of $\mathcal{G}_n^i$ while N is the number of pels in $\mathcal{A}_n^i$. D_8 flags Ω_n^i as reflection if

$$D_8 = d_8 > T_8 \tag{6.18}$$

where T_8 is a threshold. Figure 6.6 shows the ROC of D_8 as generated from processing SynRef2. Here D_8 is evaluated at $T_8 = [4.9038 \times 10^{-6} : 10^{-5} : 6.3490 \times 10^{-4}]$.

6.2.1.8 Motion Field Divergence (Detector D_9)

The motion field divergence d_9 for the examined patch Ω_n^i is defined here as follows

$$d_9 = \text{DFD(x,y)} \times (\|\text{div}(S_n(x,y))\| + \|\text{div}(S_{n+1}(x,y))\|)/2 \tag{6.19}$$

where

$$\text{div}(S_{(\cdot)}(x,y)) = \frac{\partial S_{(\cdot)}(x,y)}{\partial x} + \frac{\partial S_{(\cdot)}(x,y)}{\partial y} \tag{6.20}$$

Here $S_n(x,y)$ is the horizontal and vertical motion components of Ω_n^i while $S_{n+1}(x,y)$ is the horizontal and vertical motion components of Ω_{n+1}^i. Motions are calculated using a simple single motion block matching. DFD is the motion compensated Displaced Frame Difference for Ω_n^i. DFD is set to the minimum of the forward and backward DFDs. $\text{div}(S_{(\cdot)}(x,y))$ at the examined frame is set to the minimum of the forward and backward divergence. We use the average of $\text{div}(S_n)$ and $\text{div}(S_{n+1})$ to reject possible motion blur generated by unsteady camera motion.

DFD and div(·) take high values in regions containing reflections as single motion estimators fail in such regions. Hence, the reflection detector D_9 proposed here flags a region as reflection if

$$D_9 = d_9 > T_9 \tag{6.21}$$

where T_9 is a threshold. Figure 6.6 shows the ROC of D_9 as generated from processing SynRef2. Here D_9 is evaluated at $T_9 = [0:1:139]$. As shown in figure 6.6, $[D_1, D_6, D_9]$ are the three best performing detectors among the 9 presented weak detectors.

6.2.2 Strong Detector Generation with Adaboost (D_s)

The strong detector D_s is expressed as a linear combination of several weak classifiers. The weak classifiers are obtained from D_{2-8} where each weak classifier operates at a threshold T. This means D_s can contain several weak classifiers from (say) D_2, however each one will operate at a different threshold. We found that not including [D_1,D_9] in D_s generates better detection results than when included. The strong detector D_s operates at threshold T_s and is expressed as follows

$$D_s = \left(\sum_{k=1}^{N} \mathcal{W}_{(V(k),T_k)} D_{(V(k),T_k)} \right) > T_s \tag{6.22}$$

where

$$D_{(V(k),T_k)} = \begin{cases} d_{V(k)} < T_k \text{ for } V(k) = 2,3,4,5 \\ d_{V(k)} > T_k \text{ for } V(k) = 6,7,8 \end{cases}$$

Here N is the number of weak classifiers used in forming D_s and $V(k)$ is a function which returns a value between $2-8$ to indicate which detectors from D_{2-8} are used. k indexes the weak classifiers in order of their importance as defined by the weights $\mathcal{W}$ while T_k is the operating threshold of the k^{th} weak classifier. $\mathcal{W}$ and T_k are learned through Adaboost [89] by training D_s on the artificially created database SynRef2.

Figure 6.7 shows the reflection classification error of D_s as generated from processing SynRef2. The classification error ϵ is defined as follows

$$\epsilon = \sum_i w_i |\tau_i - h_i| \tag{6.23}$$

Here i indexes the training samples. Ground-truth detection labels are denoted by τ where $\tau_i = 1$ denotes reflection in sample i while $\tau_i = 0$ denotes the absence of reflection. h is the detection generated by Adaboost which is defined in a similar way to τ. w_i are weights generated by Adaboost [89]. As shown in figure 6.7 the classification error reduces as more weak classifiers are used in generating D_s. The classification error starts to saturate after using 15 classifiers. Hence we use 15 weak classifiers in forming D_s. Table 6.1 shows $\mathcal{W}$ and T of the weak classifiers used in generating D_s. $\mathcal{W}$ of the 15^{th} detector is close to zero which shows any more detectors will not provide much useful information in the detection process. Figure 6.6 shows the ROC

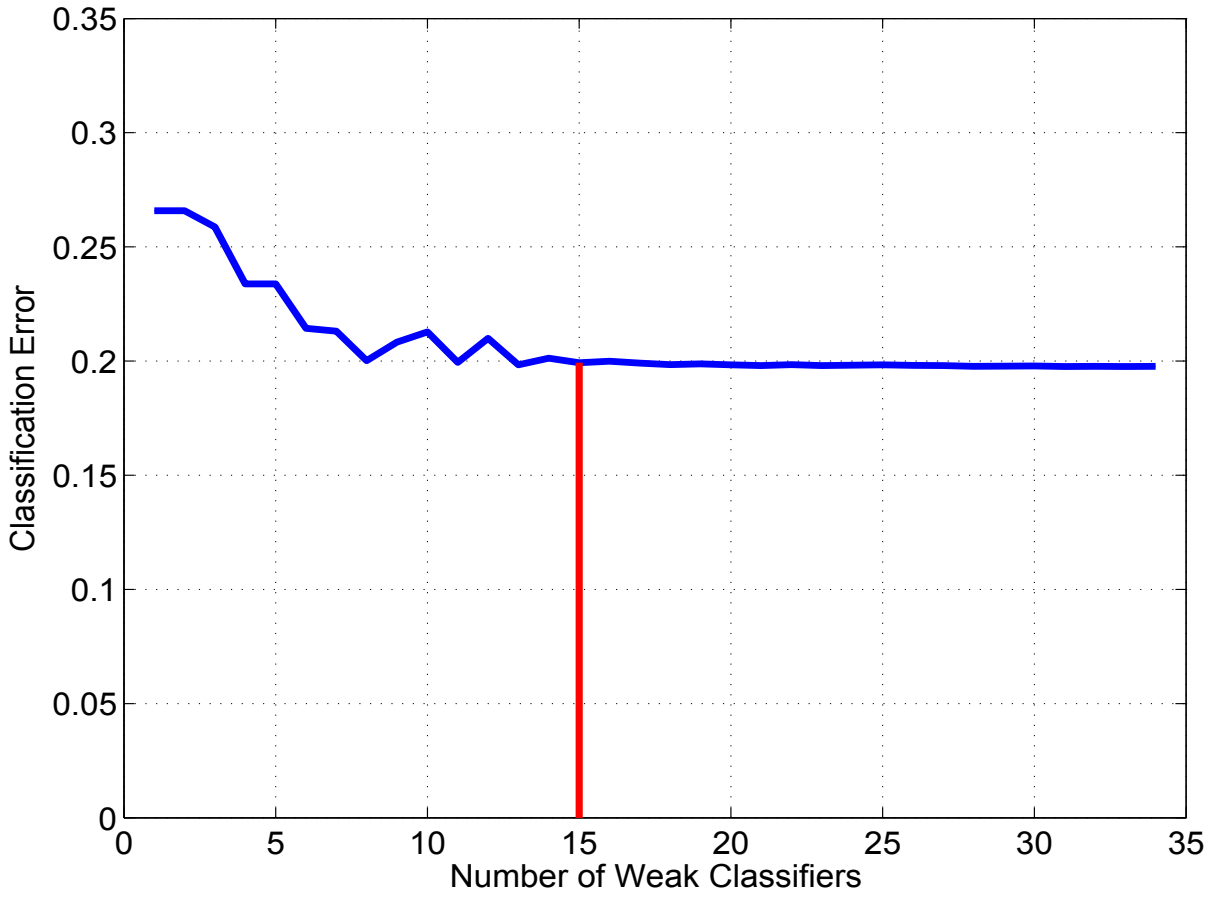


Figure 6.7: *Reflection classification error of D_s. Classification error reduces as more weak classifiers are used in generating D_s. The red vertical bar denotes the point where 15 weak classifiers are used in generating D_s.*

of D_s as generated from processing SynRef2. Here D_s is evaluated at $T_s = [0 : 0.01 : 5.2]$. As shown D_s outperforms all the weak detectors D_{1-9}.

	D_6	D_5	D_8	D_4	D_3	D_7	D_5	D_2	D_4	D_4	D_8	D_4	D_5	D_2	D_4
$\mathcal{W}$	1.15	0.71	0.48	0.53	0.38	0.26	0.2	0.22	0.17	0.17	0.2	0.1	0.12	0.09	0.1
T	0.31	0.33	0.49e-5	6.2	0.96	2.2	0.07	0.63	8.1	2.1	4.9e-6	2.2	0.48	0.37	10.1

Table 6.1: *Weights $\mathcal{W}$ and operating thresholds T for the weak classifiers used in generating D_s.*

6.2.3 Combined Detection (D_c)

The final combined detector D_c is formed by consensus among the three detectors $[D_1, D_s, D_9]$ as follows

$$D_c = (D_1 < T_1) \times (D_s > T_s) \times (D_9 > T_9) \tag{6.24}$$

Here $[T_1, T_s, T_9]$ are parameters set to $[0, 3.15, 10]$ in all examined sequences. This configuration was found empirically by processing several sequences.

Figure 6.8 (first row) shows reflection detection on three consecutive frames as generated by D_c. We call this approach ***Fea**ture **P**oint **A**nalyses for **R**eflection **D**etection* algorithm i.e. FEAPARD. The actor in figure 6.8 is moving his hands quickly generating pathological motion. The right and left picture frames (shown in yellow and white respectively) contain reflections while correct detections are shown in green. FEAPARD was able to detect some of the reflections in the right picture frame (shown in yellow). However, detections are not temporally consistent, the full picture frame is not detected and false alarms are generated (shown in red) near the actor's hand. In addition, FEAPARD failed to detect the reflection on the left picture frame (shown in white) as there are few KLT trajectories there. This rarely happens in most of the examined sequences.

In the next section we show how to reject false detections by imposing spatial smoothness on the generated results. We then show how to generate temporally consistent detection by incorporating temporal smoothness in the solution. Last, we show how to detect possible missed reflections by slight manual intervention in regions where there are few KLT trajectories.

6.2.4 Incorporating Spatial Smoothness

We reject false detections in FEAPARD by imposing spatial smoothness on the generated detection mask. We call this *Feature Point Analyses for Reflection Detection - Spatial* algorithm i.e. FEAPARD-S. We use thresholding by hysteresis. The idea here is that we start detection at high values of $[T_1, T_s, T_9]$. This generates detection points with high confidence. We then gradually reduce the thresholds and reject new detections that are spatially far from the earlier detections. We repeat this process until we reach a low value of $[T_1, T_s, T_9]$. We divide this process into two main stages. The first initializes the detection maps while the second refines the detection maps. We discuss each stage in detail next.

6.2.4.1 Detection Map Initialization

1. Let $T_H = [-0.4, 3.15, 10]$ denote the highest configuration for $[T_1, T_s, T_9]$.

2. Set the system thresholds T to T_H.

3. Estimate the FEAPARD solution of equation 6.24 using T_H. Denote the detected points by $\mathcal{M}_h$.

4. For every point in $\mathcal{M}_h$, calculate its euclidean distance Δ_{euc} with every other point.

5. For every point in $\mathcal{M}_h$, calculate its geodesic distance Δ_{geo} with every other point. Geodesic distance has been used in image processing for some time now [67]. Unlike

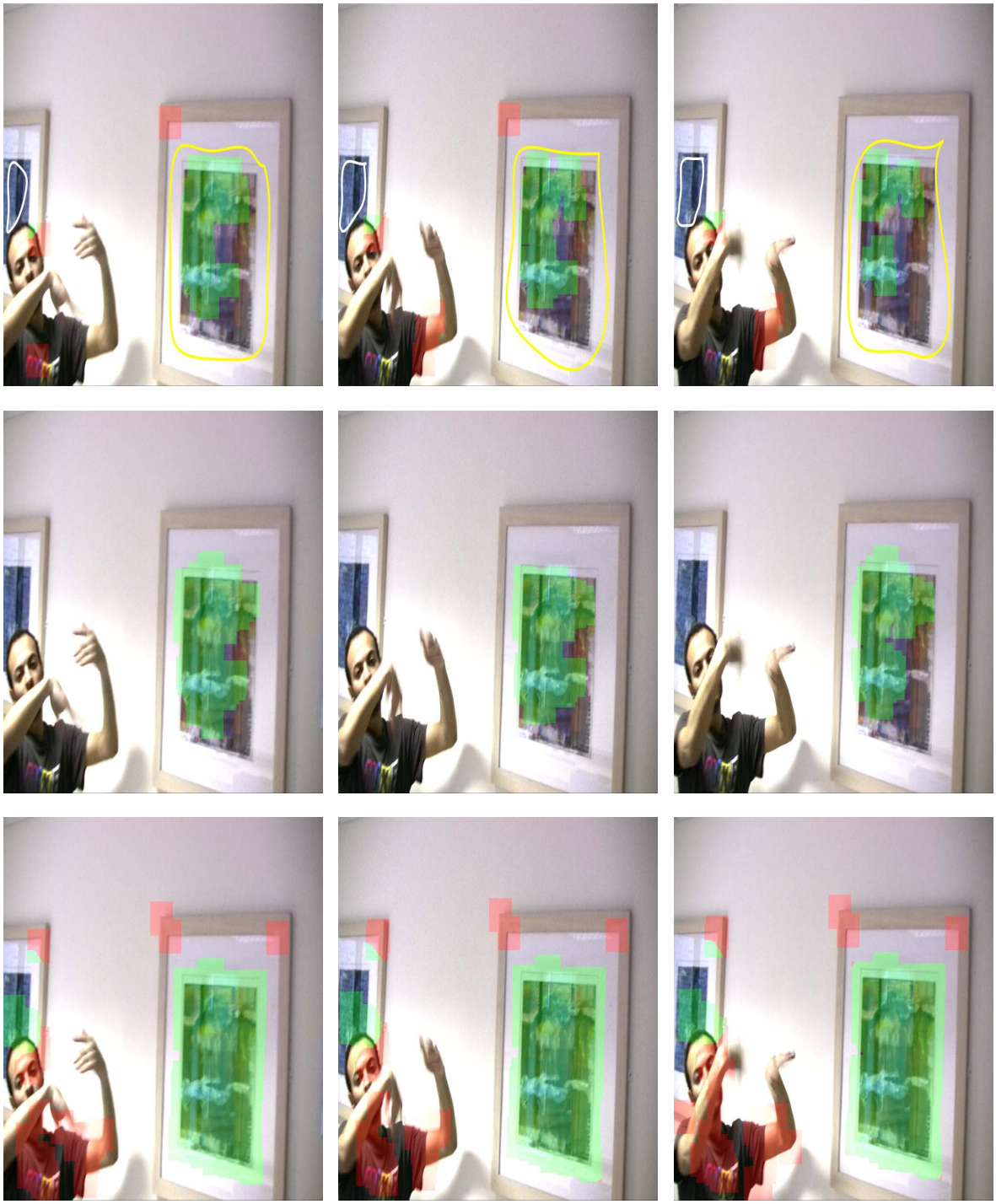


Figure 6.8: *From top; Reflection detection using FEAPARD, FEAPARD-S and FEAPARD-S with no Geodesic Distance. Ground truth reflections are shown in yellow and white (top row), correct detection shown in green and false detection is shown in red. Here FEAPARD failed to detect the region shown in white (in the left picture frame, first row) as it contains few feature point trajectories.*

the Euclidean distance, the geodesic distance takes into account the topology of the image. Points that are within the same flat region are close to each other while points that are separated by strong image gradients are far away. Points that lie on reflections have low geodesic distance between each other as reflections are areas of low gradients. For computational speed we resize the image by a factor of 50. Even though this removes many image details, results show that the geodesic distance improves detections.

6. A point in $\mathcal{M}_h$ is flagged as detected if the sum of Δ_{geo} and Δ_{euc} between the two closest points is less than 0.0025 and 4 respectively. Those values are set empirically and are fixed in the examined sequences. If there is only one point close to the examined point, we discard the examined point from the detection mask.

7. Flag a trajectory point as reflection if it lies in a region (indicated by Ω) that is detected as reflection. This extends correct detection in space which generates spatially consistent detections.

The detection map generated from Steps 1-7 is denoted by $\mathcal{M}_i$.

6.2.4.2 Detection Map Refinement

1. Let $T_L = [0, 3.15, 10]$ denote the lowest configuration for $[T_1, T_s, T_9]$.

2. Set $T = T - [-0.01, 0, 0]$.

3. Estimate the FEAPARD solution of equation 6.24 using T. Denote the detected points by $\mathcal{M}_h$.

4. For every detection point that exists in $\mathcal{M}_h$ and does not exist in $\mathcal{M}_i$, calculate its euclidean distance Δ_{euc} and geodesic distance Δ_{geo} with every detection point in $\mathcal{M}_i$. Again we resize the examined images by a factor of 50 for computational speed.

5. Flag a point in $\mathcal{M}_h$ as correct detection if the sum of its geodesic and euclidean distance with the two closest points in $\mathcal{M}_i$ is less than 0.0025 and 4 respectively. This generates new detections that are spatially consistent with the earlier detection $\mathcal{M}_i$.

6. Perform steps 2-5 from the 'Detection Map Initialization' stage on the points of $\mathcal{M}_h$ that are not flagged as correct detections. This keeps new correct detection points that are not spatially consistent with $\mathcal{M}_i$ and discards all the remaining points from the detection map $\mathcal{M}_h$.

7. Set $\mathcal{M}_i$ to $\mathcal{M}_h$.

8. Repeat steps 2-7 until T reaches T_L.

Figure 6.9 shows intermediate results of FEAPARD-S as T reaches three different configurations. The configurations used in figure 6.9 are (from top) $[-0.15, 3.15, 10]$, $[-0.09, 3.15, 10]$ and $[-0.07, 3.15, 10]$. Detection becomes more spatially consistent as T is lowered. Figure 6.8 (second row) shows the final detection of FEAPARD-S. False alarms of FEAPARD (shown in red, first row) are correctly removed. In addition, detection of the right picture frame (shown in yellow, first row) is now more spatially consistent than the detection of FEAPARD. Figure 6.8 (third row) shows the importance of the geodesic distance in rejecting false detections. Here we ignore the geodesic distance from the spatial smoothness and use only the euclidean distance to reject false detections. Figure 6.8 (third row) shows that ignoring the geodesic distance from the spatial smoothness leads to the generation of false detections (shown in red).

FEAPARD-S (see figure 6.8, second row) generates incomplete detections of the right picture frame (shown in yellow, first row). In the next section we show how to improve this detection by imposing temporal smoothness on FEAPARD-S.

6.2.5 Incorporating Temporal Smoothness

We impose temporal smoothness on the generated detections by adding an extra step after step (7) in the 'Detection Map Initialization' stage of FEAPARD-S. Step (7) extends the correct detections in space only. In the extra step we extend the correct detections in time as well. Basically if a point along a trajectory is detected as reflection ($l_n^i = 1$), all feature points lying along this trajectory i are then flagged as reflection ($l = 1$). We call this step temporal dilation and we call the final detection algorithm with spatio-temporal smoothness *Feature Point Analyses for Reflection Detection - Final* algorithm i.e. FEAPARD-F. For future reference we call the detection algorithm with only temporal smoothness (without spatial smoothness) *Feature Point Analyses for Reflection Detection - Temporal* algorithm i.e. FEAPARD-T. Figure 6.10 (first row) shows the result of FEAPARD-F. Reflection detections in the right picture frame (on the wall) are now more consistent in both space and time than the FEAPARD-S detection (see figure 6.10, second row).

6.2.5.1 An optional Key Mask Propagation Step

To increase spatio-temporal consistency further, an optional temporal refinement step is performed to fill missed detection holes as the one shown in yellow in figure 6.10 (first row). The idea is to combine detection information from a frame where the reflection is completely detected with detection in the current frame. A key detection frame K is set to the largest connected detection mask in the examined sequence. We propagate this mask to the remaining frames of the examined sequence. The motion between the key frame K and a frame f in the examined sequence is modeled by a 2D affine transformation. A pel $[x_k, y_k]$ in the key frame where $l = 1$

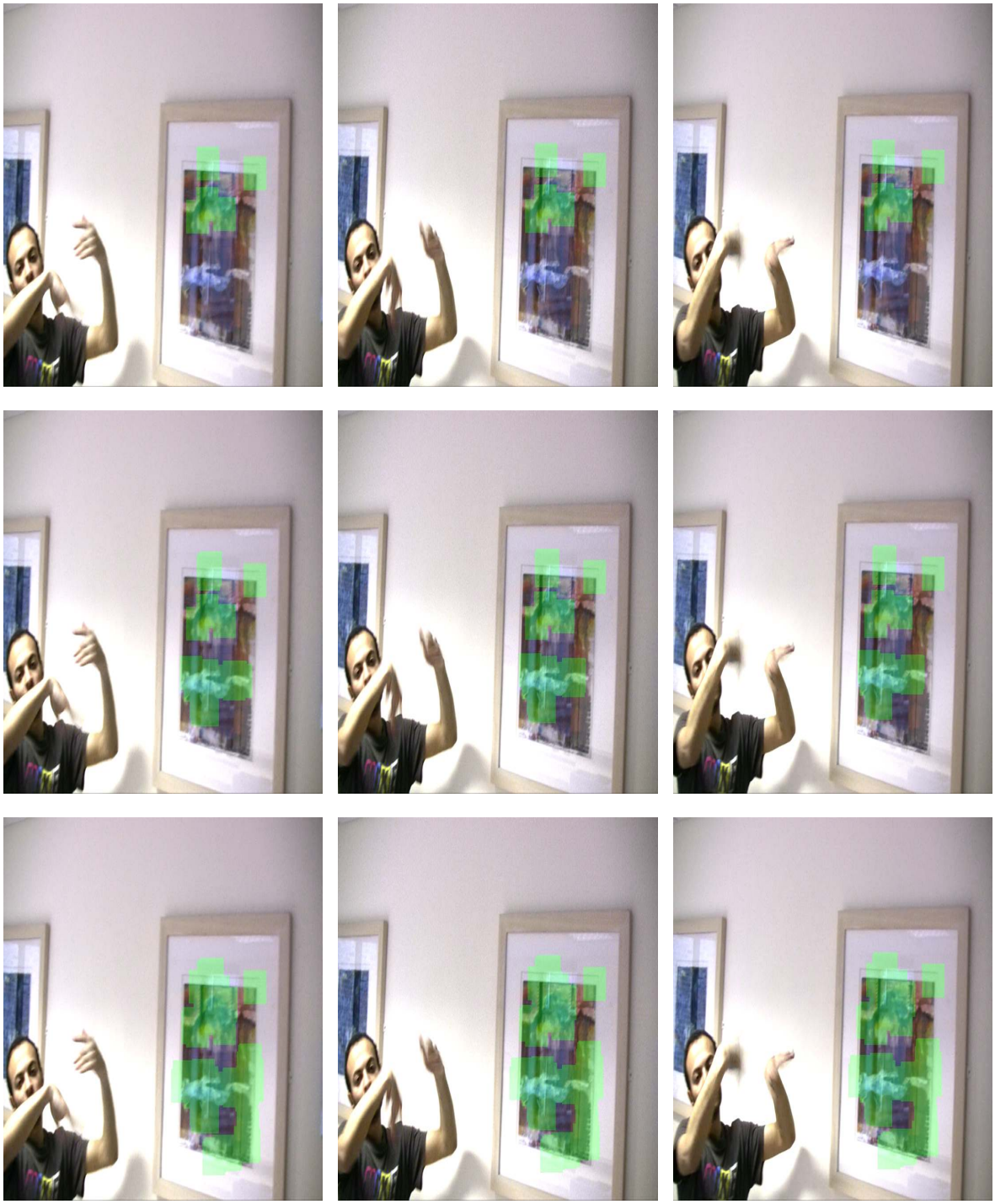


Figure 6.9: *From top; Three intermediate stages of FEAPARD-S at* $T = [-0.15, 3.15, 10]$, $[-0.09, 3.15, 10]$ *and* $[-0.07, 3.15, 10]$ *respectively. Correct detection are shown in green. Detection becomes more dense as* T *is lowered. Here no false detections are generated.*

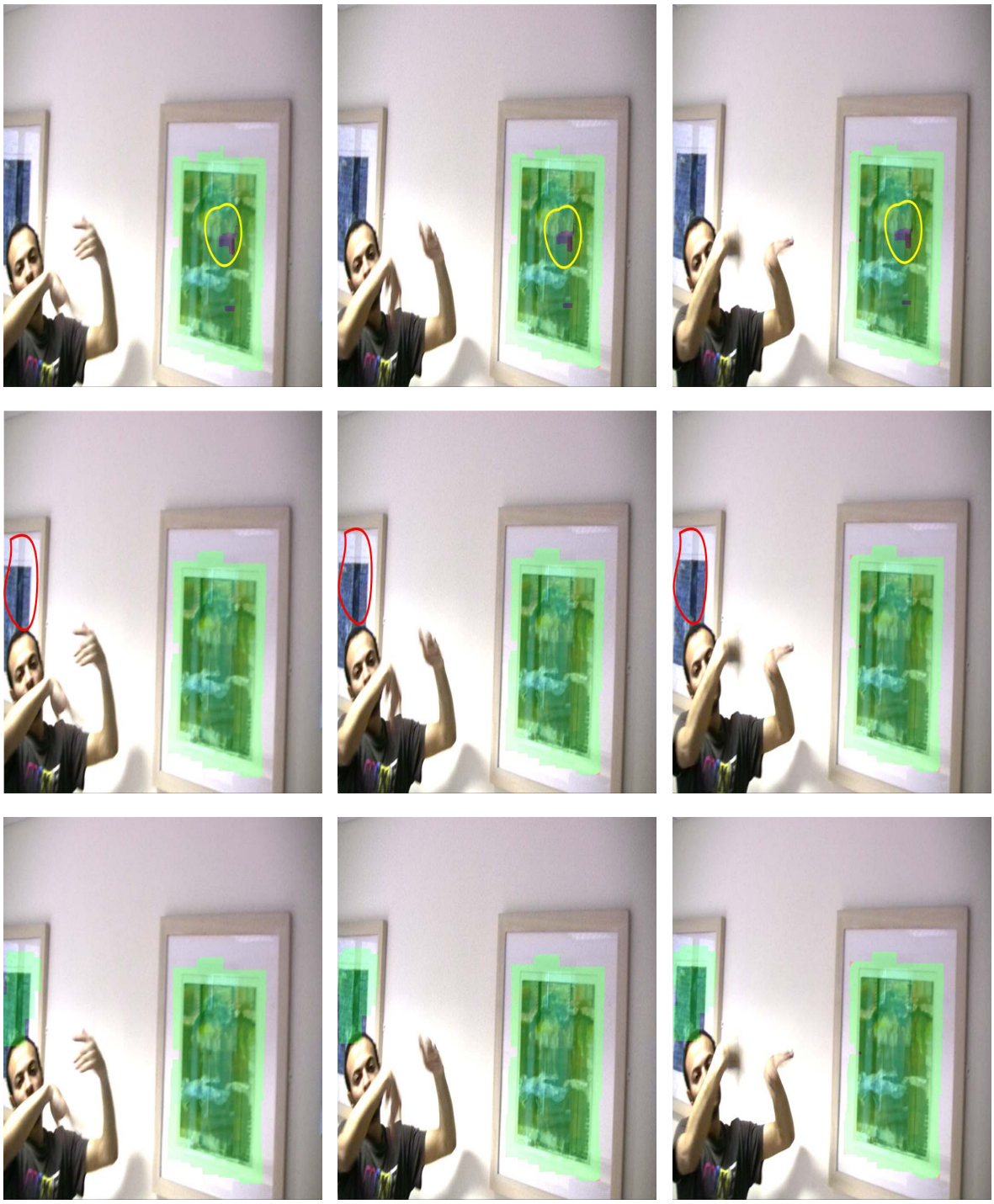


Figure 6.10: *First two rows; Correct reflection detection (shown in green) using (from top) FEAPARD-F and FEAPARD-F with Key-Mask propagation. Last row; Recovering missed detections (shown in red, second row) using slight manual intervention. The user-supplied detection mask used here is shown in figure 6.12*

is mapped to pel $[x_f, y_f]$ in frame f using the following relation

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} \tag{6.25}$$

Here $[a_{11}, a_{12}, a_{13}, a_{31}, a_{32}, a_{33}]$ are the 2D affine transformation parameters to be estimated. $[x_k, y_k, x_f, y_f]$ are obtained from the KLT trajectories and the affine parameters are estimated using least square fitting. The key mask is propagated to frame f using the estimated motion parameters.

A pel in frame f is flagged as reflection if it is detected as reflection by FEAPARD-F and if it coincides spatially with the key mask which is temporally propagated from frame K. This ensures the removal of false detections from the FEAPARD-F estimate (if any) but may however cause some true reflection sites to be missed. To solve that problem we apply 'temporal dilation' on the generated detections as explained in Section 6.2.5. This generates denser maps and detects missed sites. Figure 6.10 (second row) shows the final detection with this key mask propagation step. The region shown in yellow (first row) is now detected. Figure 6.11 shows the main components of our reflection detection technique.

FEAPARD-F failed to detect regions that have few feature point trajectories (see figure 6.10 second row, shown in red). We propose to detect such regions (if necessary) using slight manual intervention.

6.2.6 Slight Manual Intervention for Detecting Missed Sites

Detecting regions that contain few feature points requires slight manual intervention. The user supplies rough hand-drawn masks for the missed regions. In the examined sequences an average of 4 user masks are supplied for every 50 frames. Those masks should encompass as much of the missed reflections in the examined frame (see figure 6.12). All feature points encompassed in the supplied masks are flagged as reflection. Labels are then extended in space and time using the temporal dilation stage discussed in section. 6.2.5. We do not perform any further spatial or temporal refinement. Figure 6.10 (bottom line) shows the ability of recovering missed detections using this slight manual intervention. For the examined sequence we supply an average of one hand-drawn reflection mask every ten frames. The mask used for the three consecutive frames of figure 6.10 is shown in figure 6.12.

6.3 Results

17 sequences altogether containing 1012 frames of size 576×720 are examined (see figure 6.13-6.16). Some of the examined sequences contain pathological motion and/or occlusion. No author has attempted to detect reflections in an explicit form, hence we compare our technique FEAPARD-F against two straightforward reflection detectors named DFD and SHARPNESS. DFD flags a region as reflection if its motion compensated DFD is higher than a certain threshold

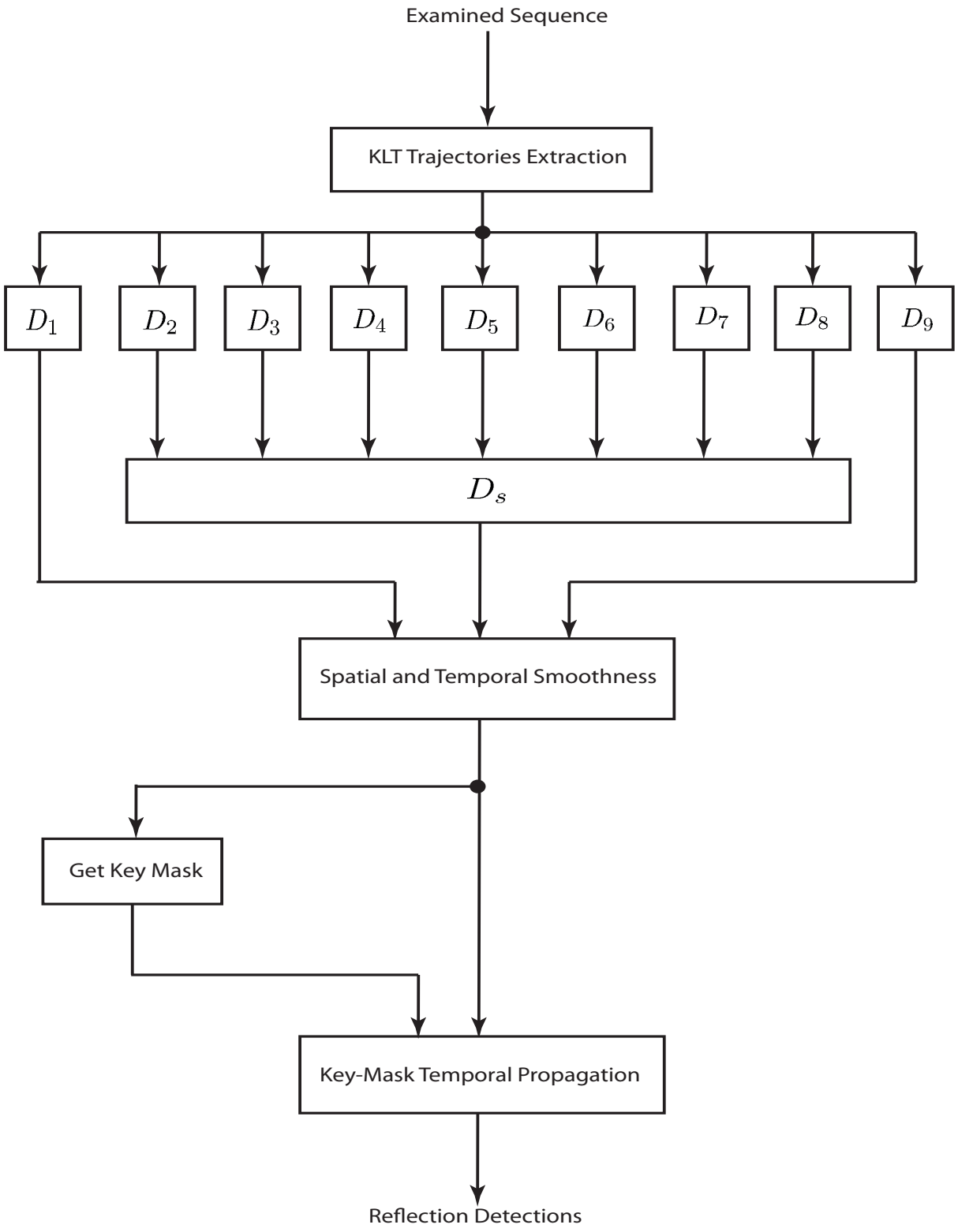


Figure 6.11: *Main Components of the FEAPARD-F System.*

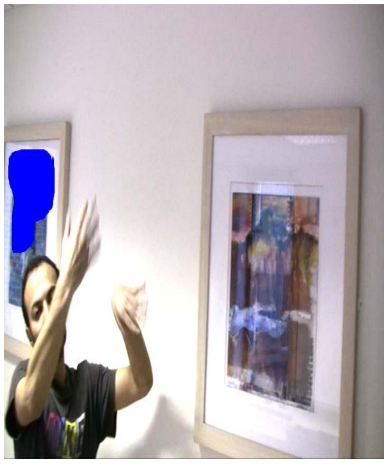


Figure 6.12: *User supplied detection mask (shown in blue) for the reflection shown in red in figure 6.10 (second row).*

while SHARPNESS flags regions with low image contrast as reflection. Image sharpness is calculated using the Ferzil et al. approach [27] as before and both detectors process frames in blocks of 50×50 pels. DFD and SHARPNESS are easy to implement and hence they can be a popular alternative to our technique FEAPARD-F. In addition their evaluation allows the assessment of using spatial and temporal information for reflection detection. We also examine the importance of D_{1-9} and D_s in generating the FEAPARD-F detection. Last, we examine in more detail the importance of incorporating spatial and temporal information in FEAPARD-F. Results show that the geodesic distance used in the spatial information helps in rejecting false detections. In all experiments we use ground-truth detections to assess the performance of the examined techniques. Ground-truth estimates are generated by manually selecting regions of reflections.

6.3.1 Comparison with other techniques

Figure 6.17-6.22 shows detection results of FEAPARD-F, SHARPNESS and DFD on 5 out of the 17 sequences. Results for more sequences are in Appendix C (see figure 11.1-11.9). Image sequence results can be found in the accompanying DVD. In figure 6.17-6.19 and figure 6.22 the actor generates pathological motion by moving his hands quickly. Occlusion is shown in yellow in figure 6.20 (first row). We use 0.1 and 6 as threshold values for SHARPNESS and DFD respectively. SHARPNESS misclassifies flat areas as reflection and misses many sites of true reflections. DFD detects many regions containing reflections however it misclassifies regions of pathological motion and occlusion as reflection. FEAPARD-F detects reflections correctly and rejects flat areas and regions containing complicated motion. FEAPARD-F does not detect the left picture frame shown in white in figure 6.22 (first row) as this region contains few feature point trajectories. We detect this region using slight manual intervention as discussed earlier. In this

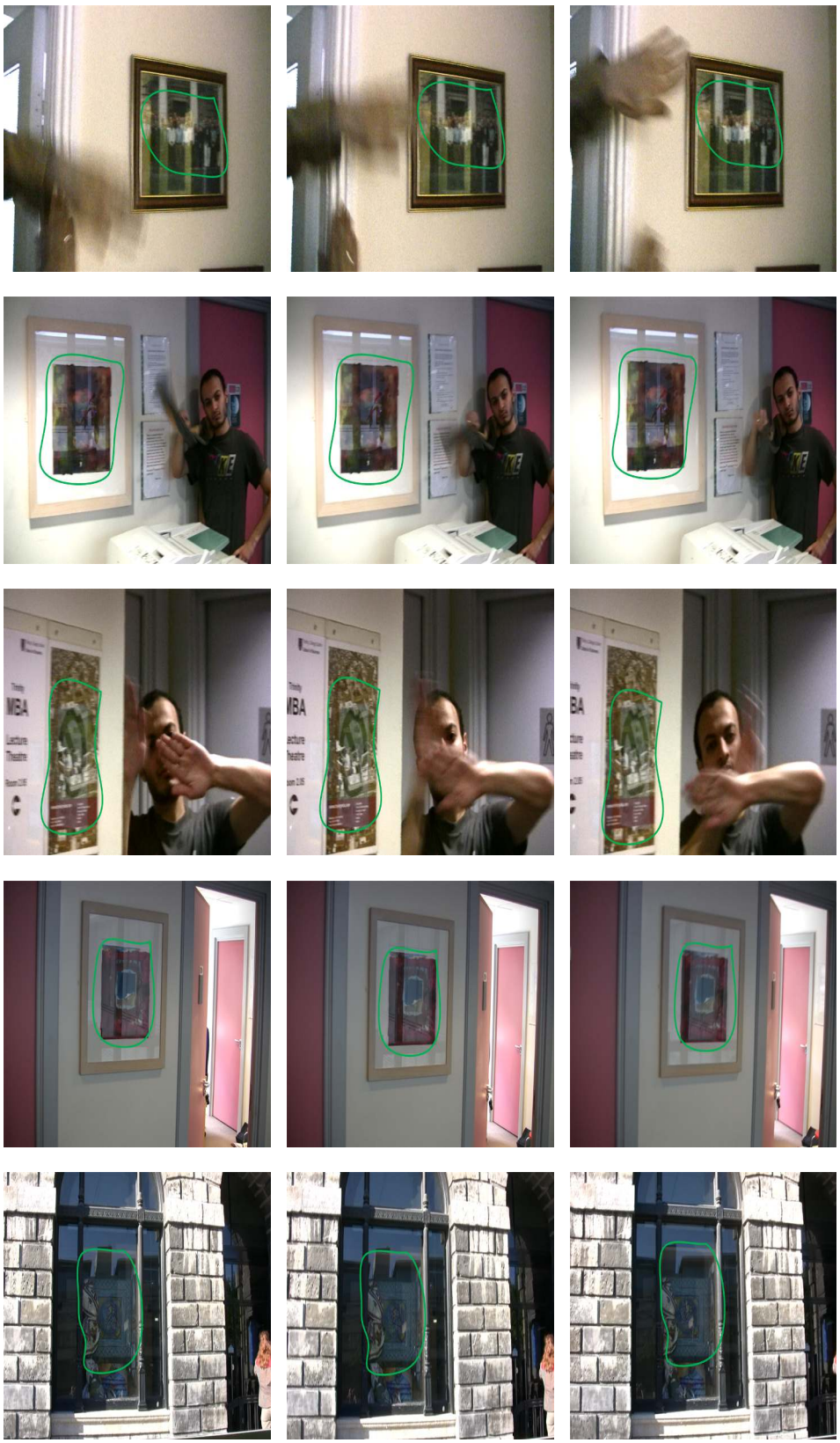


Figure 6.13: *First row; Frames 9, 10, 11 from* ***PortraitP****. Second row; Frames 13, 14, 15 from* ***SelimC****. Third row; Frames 8, 9, 10 from* ***SelimH****. Fourth row; Frames 11, 13, 15 from* ***PortraitA****. Fifth row; Frames 8, 12, 16 from* ***BuildOnWind1****. Reflections are shown in green. In the first three sequences the actor is creating pathological motion by moving his hands quickly.*

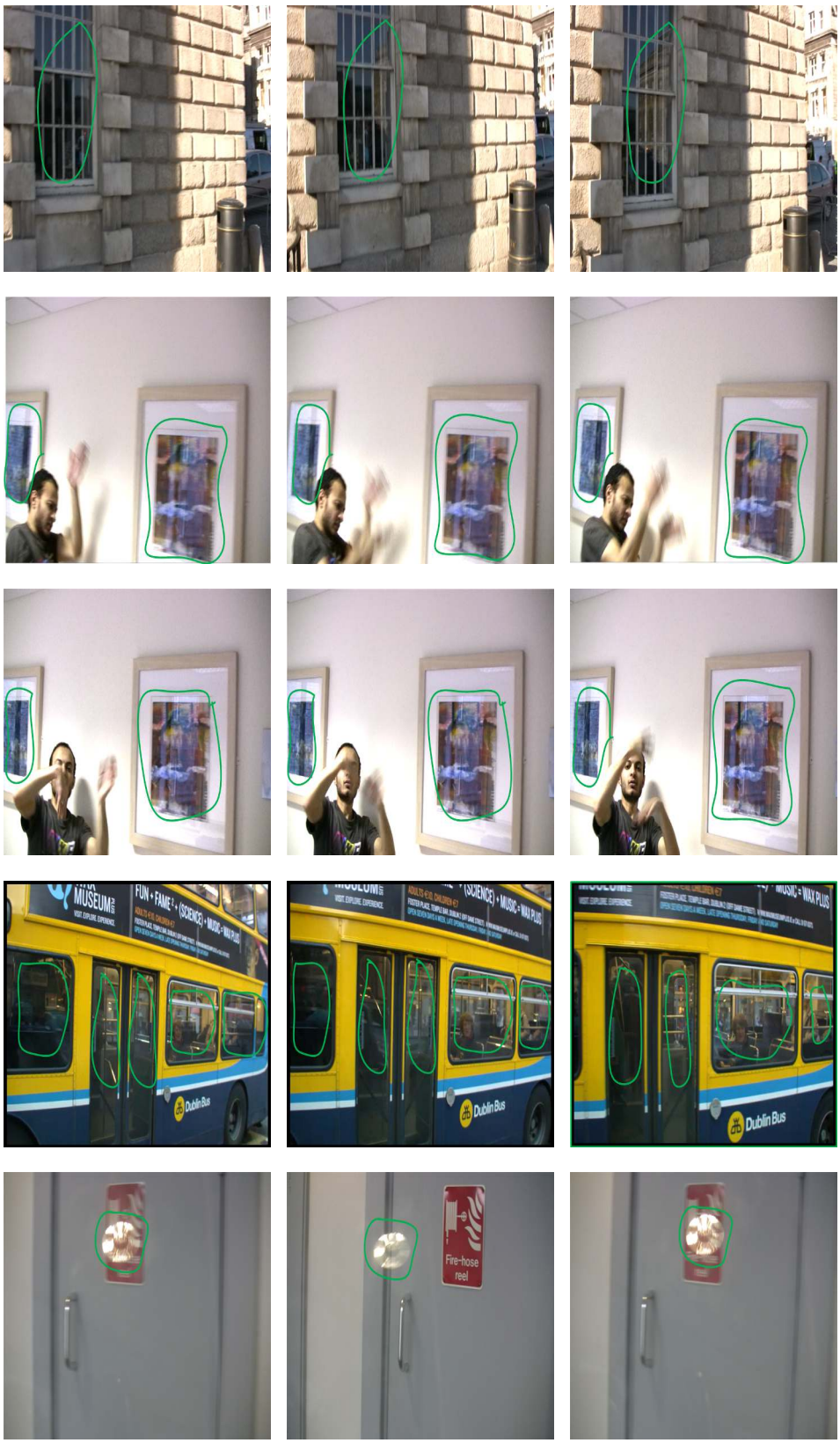


Figure 6.14: *First row; Frames 30, 40, 50 from* ***BuildOnWind2****. Second row; Frames 23, 24, 25 from* ***SelimK1****. Third row; Frames 28, 29, 30 from* ***SelimK2****. Fourth row; Frames 25, 35, 45 from* ***Bus****. Fifth row; Frames 8, 16, 30 from* ***Bulb****. Reflections are shown in green. In the the second and third sequences the actor is creating pathological motion by moving his hands quickly. The motion of the bus in the fourth sequence creates a projective distortion.*

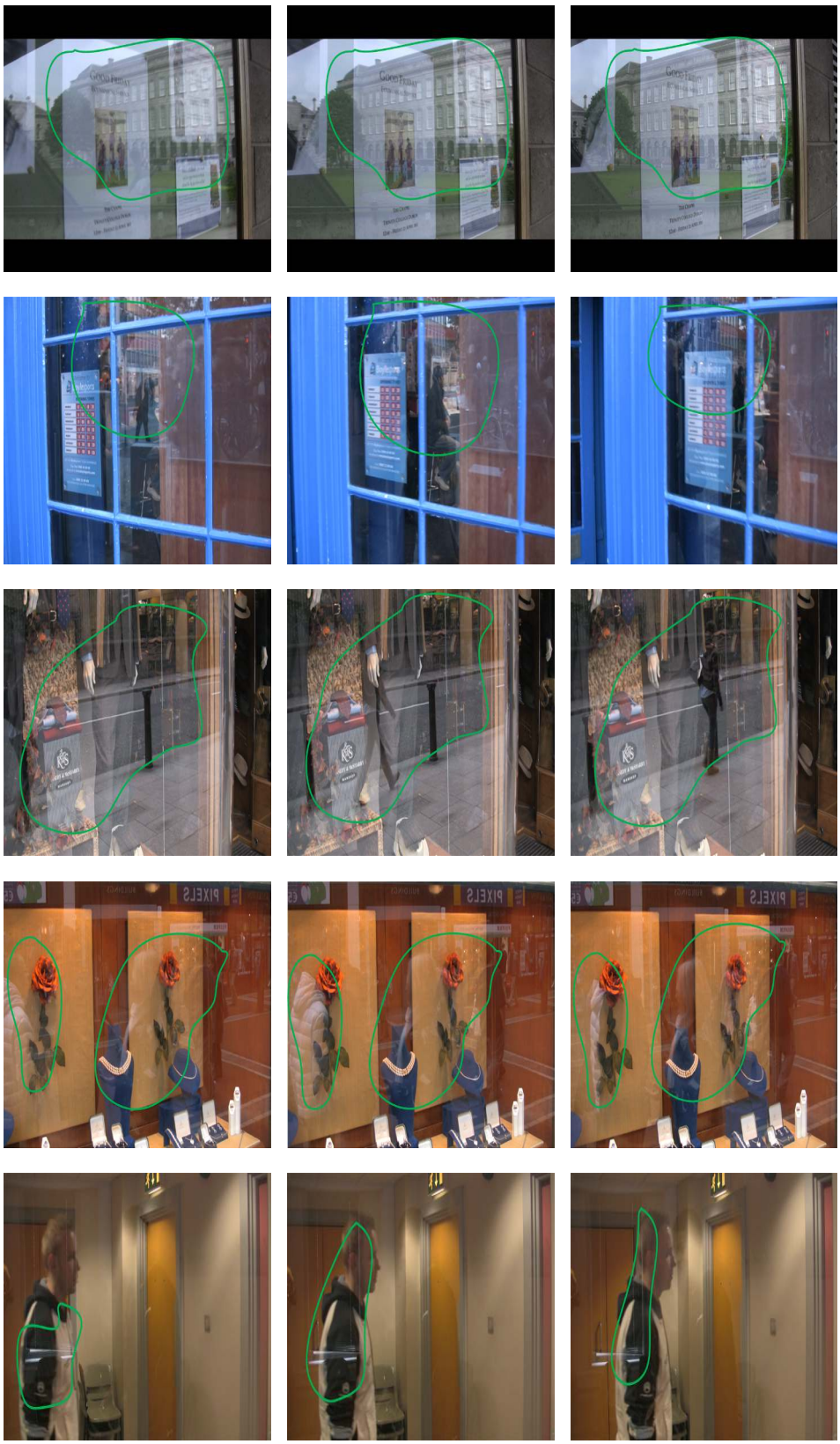


Figure 6.15: *First row; Frames 10, 20, 30 from* ***BuildOnWind3****. Second row; Frames 15, 25, 35 from* ***BluWind****. Third row; Frames 10, 30, 50 from* ***GirlRef****. Fourth row; Frames 50, 55, 60 from* ***RedPPL****. Fifth row; Frames 3, 7, 11 from* ***ManWalking****. Reflections are shown in green.*

Figure 6.16: *First row; Frames 30, 31, 32 from* ***CarRef****. Reflection of* ***CarRef*** *(shown in green) contain few feature points trajectories. Second row; Frames 13, 16, 29 from* ***GirlShadow****. The fast pathological motion of the girl generates shadow (shown in green).*

examined sequence we supply one manual detection mask every 10 frames. The user-supplied detections are shown in figure 6.21.

Figure 6.23 shows the ROC for FEAPARD-F, SHARPNESS and DFD for **PortraitP** and **PortraitA**. FEAPARD-F is evaluated at $T_1 = [-0.22 : 0.01 : 0]$ while $[T_s, T_9]$ are fixed to $[3.15, 10]$. SHARPNESS and DFD are evaluated at $T = [0 : 0.01 : 1]$ and $T = [0 : 1 : 20]$ respectively. ROCs for more sequences are shown in Appendix C (see figure 11.10-11.12). Figure 6.23 shows that FEAPARD-F generates a massive increase in the correct detection rate over SHARPNESS and DFD.

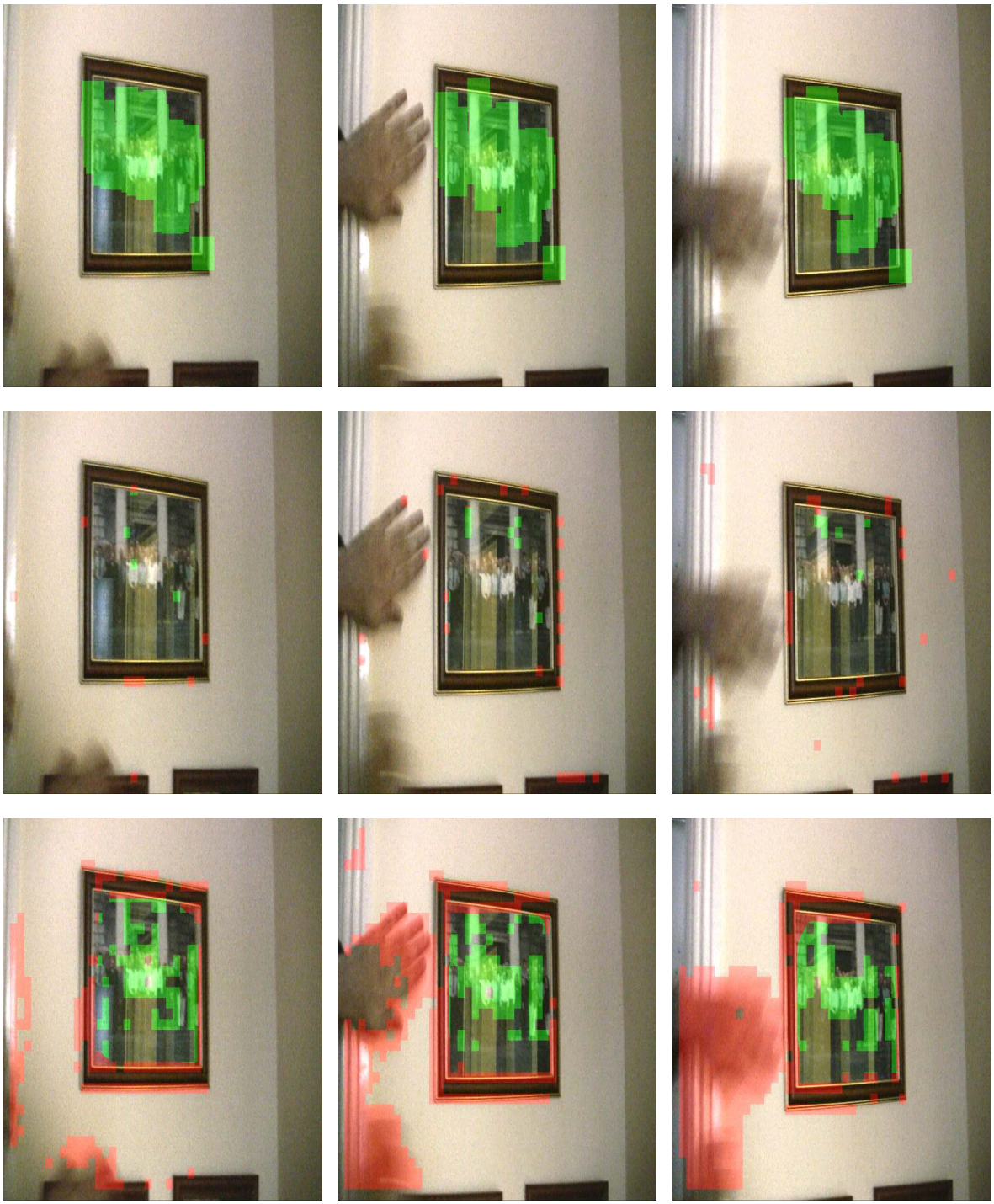


Figure 6.17: *Reflection detection on* ***PortraitP*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. The pathological motion generated by the actor's hands is rejected by FEAPARD-F but detected by DFD.*

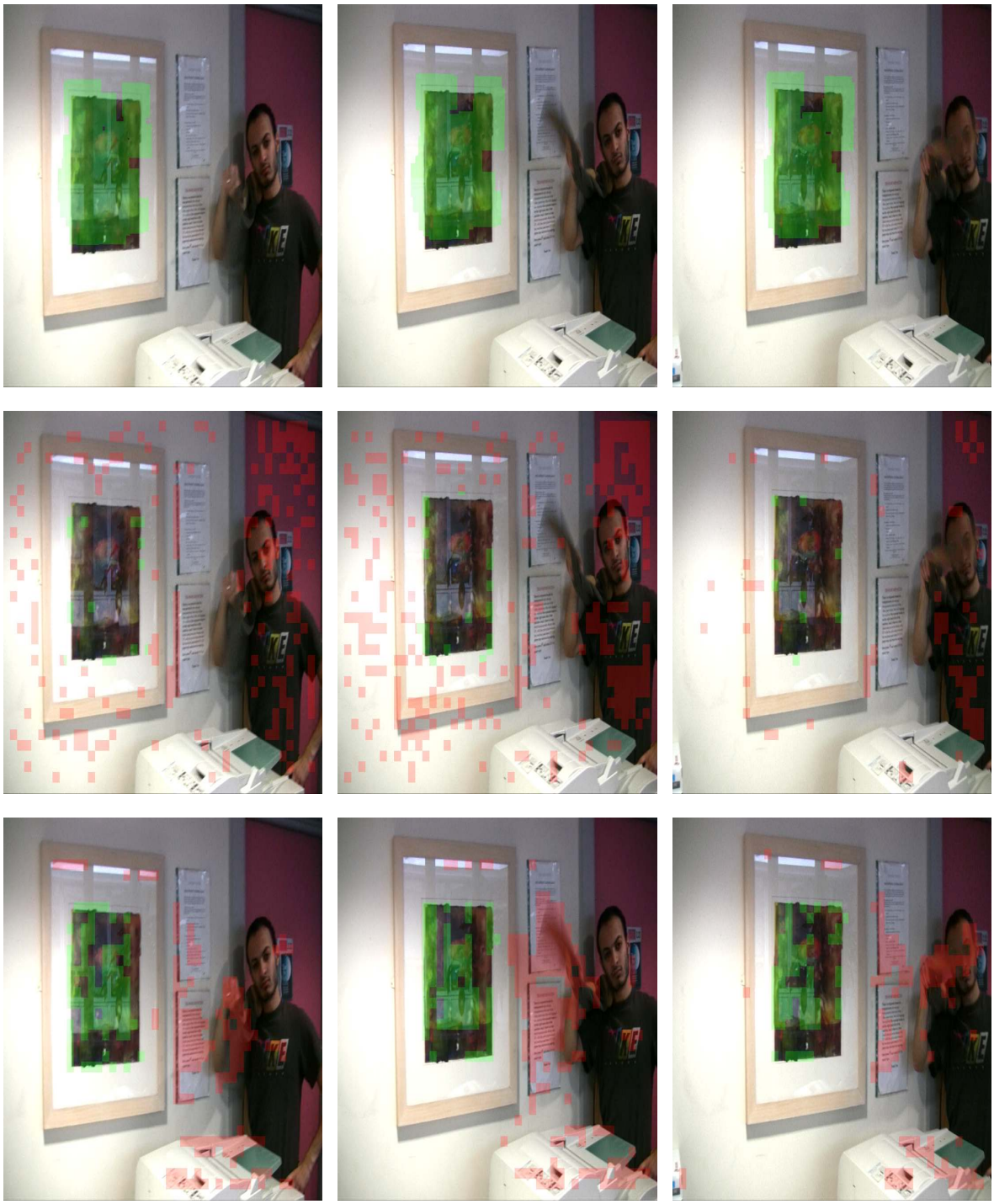


Figure 6.18: *Reflection detection on* ***SelimC*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. The pathological motion generated by the actor's hands is rejected by FEAPARD-F but detected by DFD.*

Figure 6.19: *Reflection detection on* ***SelimH*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. The pathological motion generated by the actor's hands is rejected by FEAPARD-F but detected by DFD.*

Figure 6.20: *Reflection detection on* ***PortraitA*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in blue. Some false detections are shown by blue circles (in the last row) for illustration clarity. SHARPNESS misclassifies flat areas as reflections and DFD misclassifies regions of occlusion (shown in yellow, first row) as reflections. FEAPARD-F however rejects flat areas and regions of occlusion.*

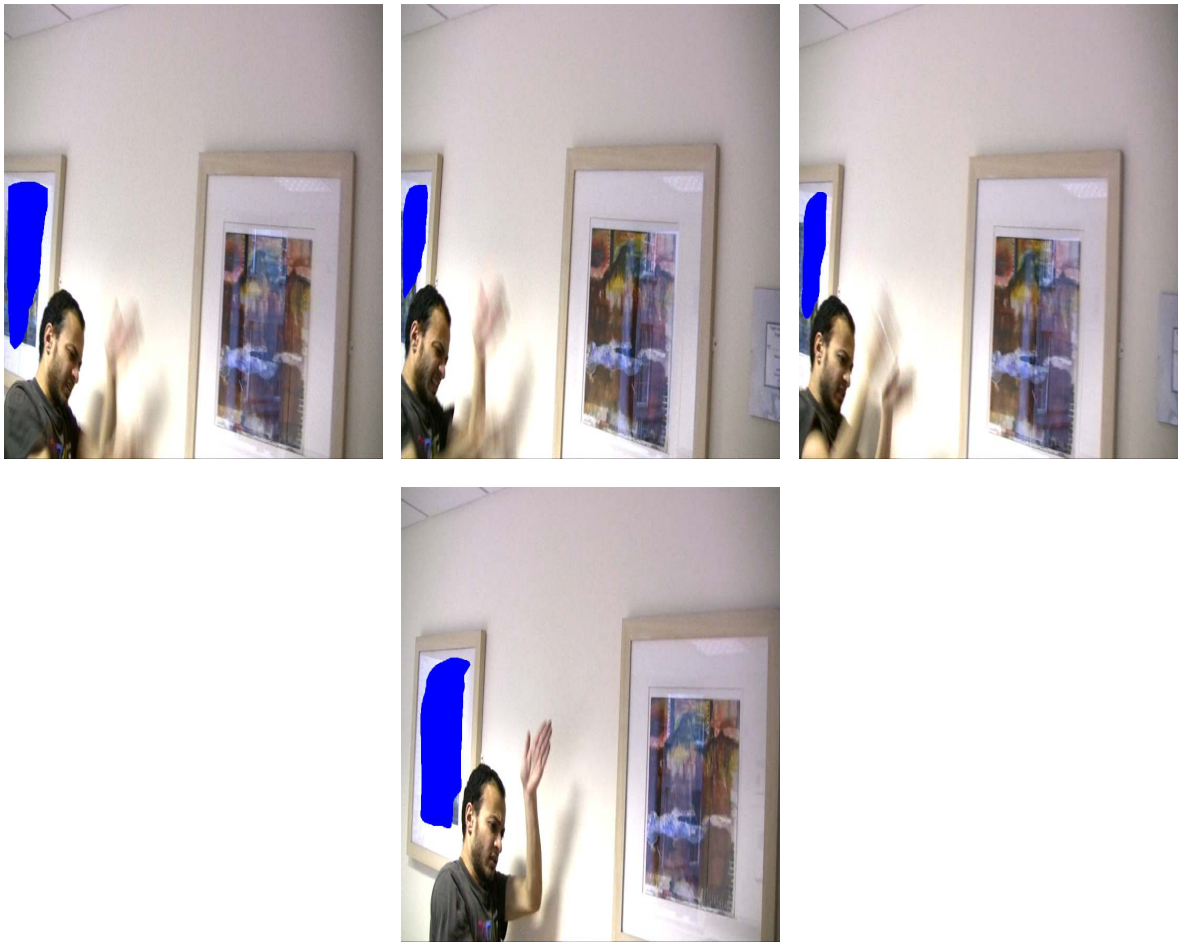


Figure 6.21: *Detection masks (shown in blue) generated manually for* ***SelimK1****. Those masks are used to detect the left picture frame in* ***SelimK1*** *(see figure 6.22, first row, shown in white).* ***SelimK1*** *contains 51 frames. Masks here are supplied at frames (in clockwise direction) 1, 10, 20 and 30 respectively.*

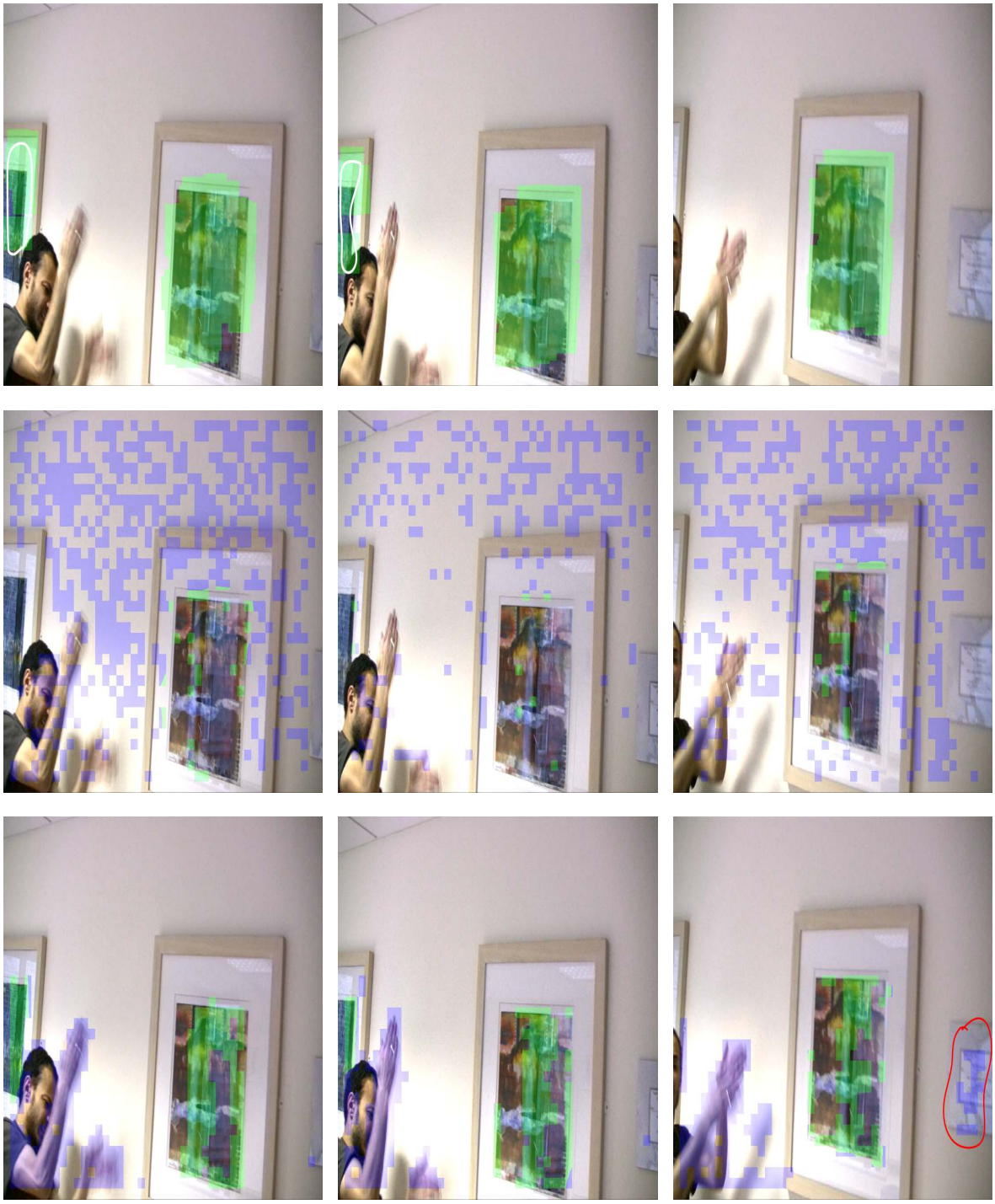


Figure 6.22: *Reflection detection on* ***SelimK1*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in blue. FEAPARD-F rejects pathological motion generated by the actor's hand. This motion however is detected by DFD. The picture frame on the left (shown in white, first row) is detected using manually supplied detection masks (see figure 6.21)*

.

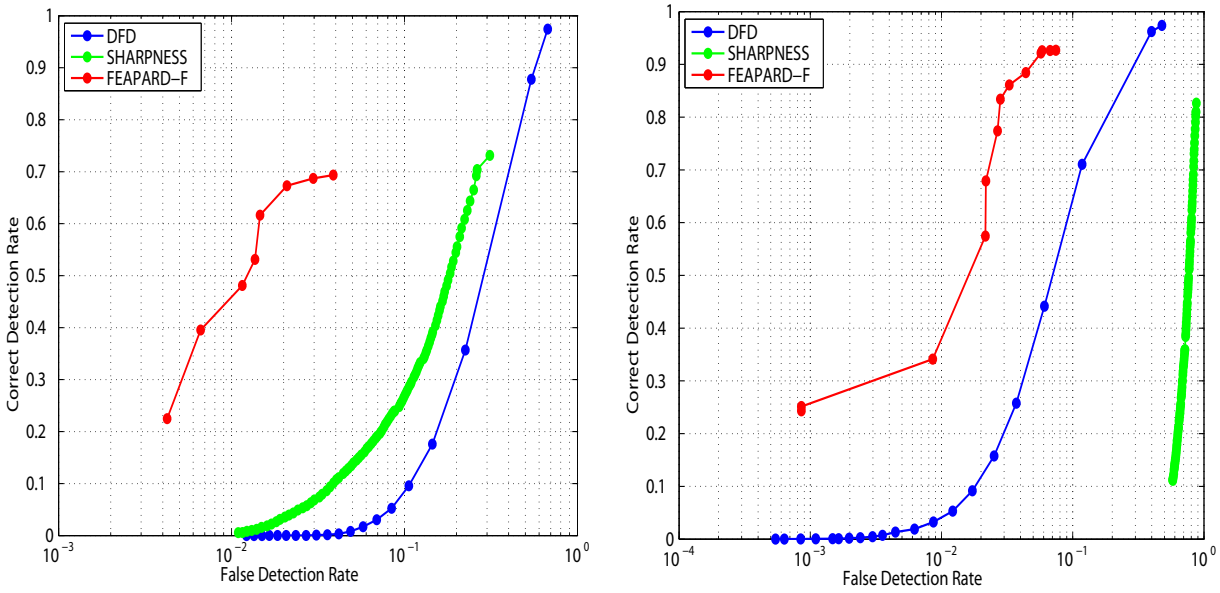


Figure 6.23: *ROC of FEAPARD-F, SHARPNESS and DFD for (from left);* ***PortraitP****, and* ***PortraitA*** *respectively. Our technique outperforms the other detectors with a massive increase in correct detection rate.*

6.3.2 Investigating the Main Components of FEAPARD-F

Figure 6.24 (first row) shows the ROC of D_{1-9} and D_s for **PortraitP** and **PortraitA**. The two main detectors used in FEAPARD-F $[D_1, D_s]$ outperform most of the remaining detectors in both sequences. ROCs for more sequences are shown in Appendix C (see figure 11.13-11.14). Figure 6.24 (second row) compares the performance D_s, D_1 and D_9 against the FEAPARD-F detection. Here however we generate the D_1 detection by removing D_s and D_9 from the FEAPARD-F technique. Detection results for D_s and D_9 are generated in a similar way. FEAPARD-F and D_1 are evaluated at $T_1 = [-0.22 : 0.01 : 0]$. We fix $[T_s, T_9]$ to $[3.15, 10]$ in FEAPARD-F. D_s and D_9 are evaluated at $T_s = [1.6 : 0.2 : 5]$ and $T_9 = [0 : 10 : 250]$ respectively. Figure 6.24 (second row) shows that the performance of D_s, D_1 and D_9 could differ from one sequence to another. However FEAPARD-F combines D_s, D_1 and D_9 in a way which optimizes detection. ROCs for more sequences are shown in Appendix C (see figure 11.15-11.16).

Figure 6.25 compares the FEAPARD-F detections of **PortraitP** and **PortraitA** against FEAPARD, FEAPARD-S and FEAPARD-T. In addition we show the FEAPARD-F detection but with ignoring the effect of the geodesic distance from the spatial smoothness. We call this technique FEAPARD-NoGeo. All techniques are evaluated at $T_1 = [-0.22 : 0.01 : 0]$ while $[T_s, T_9]$ are fixed to $[3.15, 10]$. ROCs for more sequences are shown in Appendix C (see figure 11.17-11.18). Figure 6.25 shows that FEAPARD-S generates lower false alarm rate than FEAPARD while maintaining the same correct detection rate. Imposing temporal smoothness on FEAPARD generates denser detections (see FEAPARD-T) which produce higher correct detection rate than FEAPARD (see figure 6.25). This however comes with the expense of higher false detection rate. Figure 6.25 shows that FEAPARD-F combines both spatial and temporal information of FEAPARD-S and FEAPARD-T for optimal detection. Last, figure 6.25 shows that removing the geodesic distance from FEAPARD-F (see FEAPARD-NoGeo) increases the FEAPARD-F false detection rate. This shows that the geodesic distance used in the spatial smoothness is critical in rejecting false detections.

6.4 Conclusion

This chapter has presented a technique for detecting reflections in image sequences. This problem was not addressed before. Our technique performs several analyses on feature point trajectories and generates a strong detector by combining these analyses. We generate detections that are consistent in space and time by imposing spatial and temporal smoothness on the detection field. Results show major improvement over techniques which measure image sharpness and temporal discontinuity. Our technique generates high correct detection rate with rejection of regions containing pathological motion and occlusion. The technique is fully automated for the vast majority of the examined sequences. However it may need slight manual intervention to detect missed regions that have few feature point trajectories.

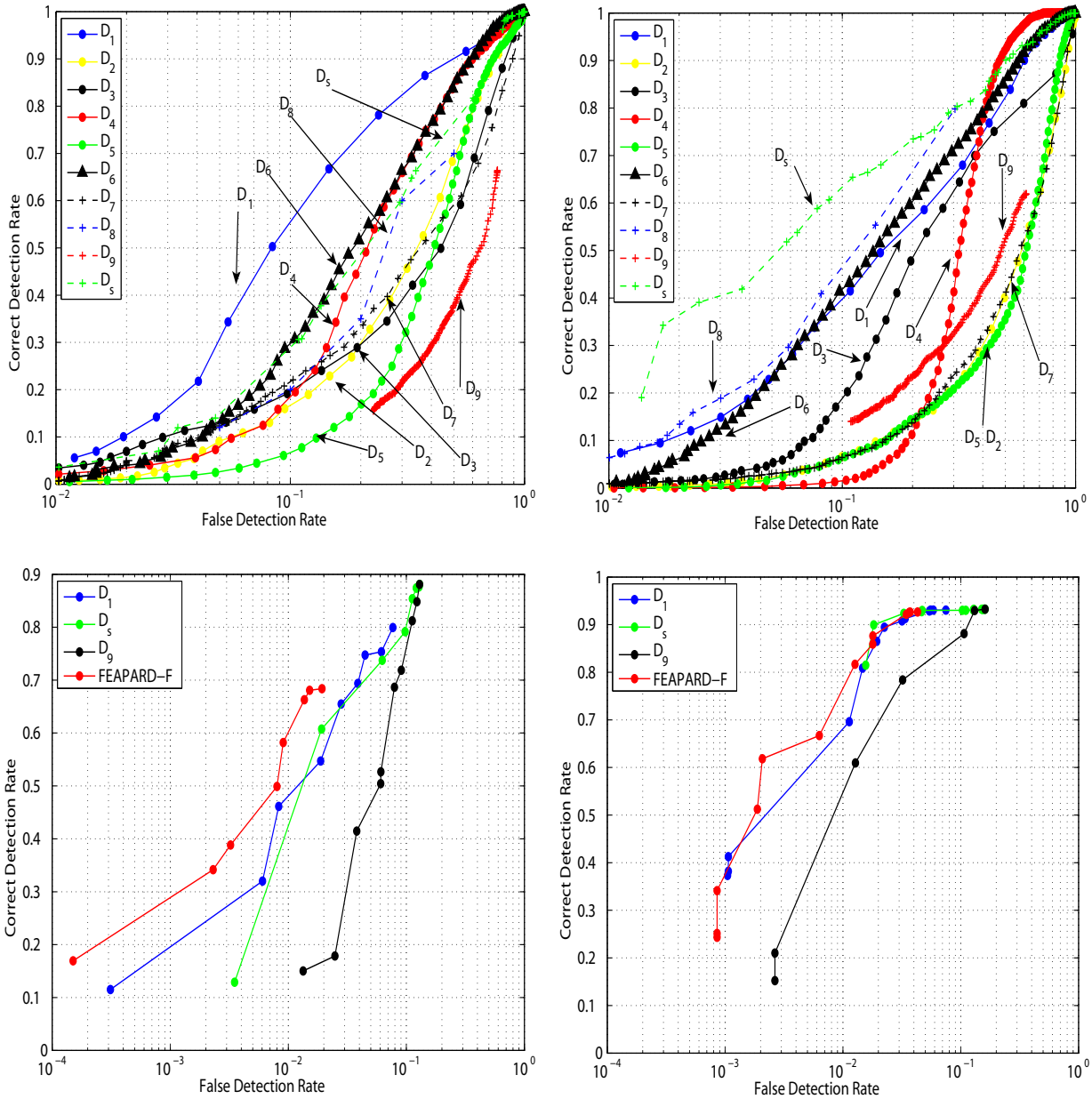


Figure 6.24: *Top; ROC of D_{1-9} and D_s for (from left);* ***PortraitP*** *and* ***PortraitA****. Either D_1 or D_s outperform most of the remaining detectors in both examined sequences. Bottom; ROC of D_1, D_s, D_9, and FEAPARD-F as generated for (from left);* ***PortraitP*** *and* ***PortraitA****. Here the detection of D_1 is obtained by removing D_s and D_9 from the FEAPARD-F technique. The detections of D_s, D_9 are obtained in a similar way. FEAPARD-F combines D_1, D_s, D_9 for optimal reflection detection.*

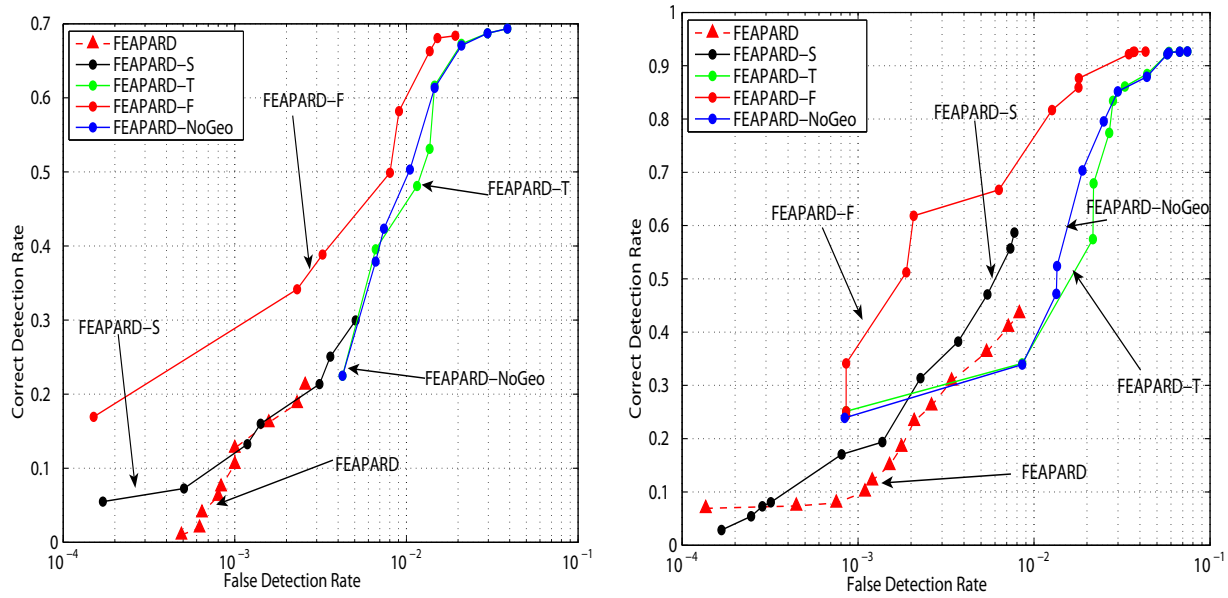


Figure 6.25: *ROC of different implementations of FEAPARD-F generated for (from left);* ***PortraitP*** *and* ***PortraitA****. In FEAPARD-NoGeo we remove the geodesic distance from the spatial information of FEAPARD-F. Results show that the geodesic distance helps in rejecting false detections and that FEAPARD-F uses both spatial and temporal information for optimal detection.*

7

Motion Estimation for Regions of Reflection

Motion estimation is important in cinema post-production for many fundamental tools e.g. dust busting, frame rate conversion, tracking and many more. Most motion estimators [48] assume that there is only one motion at a pixel site and then use the brightness consistency assumption introduced by Horn et. al [37]. Hence image brightness is assumed constant over time given small image displacement. This one motion per site idea is violated in many everyday situations e.g. shadows cast on a moving object or reflections from shiny surfaces. In those situations standard motion estimators fail and cause strange effects when the frames are compensated regardless of the application. This is because there are now at least two semi-transparent layers moving over each other generating two motion vectors per pel (see figure 7.1).

Several motion estimators have been designed to cope with these situations and traditionally there are two stages. The first stage [3,7,59,61,75,81,86,88] estimates the two motion vectors for each examined site while the second stage (motion-layer labeling) performs motion assignment to two layers [7, 81, 86]. The key issue here though, is that previous work did not explicitly involve a mixing model that explains the transparency or reflection. In Toro et al [86] the *layers* actually were support regions associated with each motion parameter. So for example in Toro et al [86], considering angiogram sequences, if there were four motions applicable to a patch, each model is associated with a separate support map indicating picture areas which were inliers and outliers with respect to the relevant optical flow equation. In Stuke et al [81] the layers referred to a classification that determined whether there were one or two motions at a site but did not actually extract the layers involved. In Vernon [88] layers are estimated however they are not used to infer motion estimates.

Figure 7.1: From left: Frames 1, 10, 20 from a sequence containing reflection of a building superimposed on paper posters. In this sequence the building is moving to the left while the posters are moving to the right. This generates two motions per pel.

Current multiple motion estimators can be classified into two main approaches. The first extends the optical flow approaches to two motions [3,59,61,75,86] while the second is based on the Fourier phase shift relationship [7,81,86,88]. Both approaches assume constant motion over at least three frames. Vernon [88] assumes constant motion over five frames. As a result current techniques cannot handle temporally active motions arising due to camera shake or acceleration. In addition, optical flow approaches can only handle small image displacements.

This chapter takes a more complete approach to the multiple motion estimation problem. We take the novel approach of exploiting a model for the observed image which is a mixture of two layers that each move according to some motion trajectory over several frames. The multiple motion estimation problem is articulated as a joint motion and layer separation problem through a Bayesian framework. Unlike Vernon [88], we use the estimated layers to infer the underlying motions. Our technique does not assume constant motion over a block of frames. The computational burden of the joint MAP solution for layers and motion is eased by generating candidates for motion (using KLT tracks [85]) and layers (using [94]) by pre-processing the observed sequence. These candidates are then evaluated within the probabilistic framework to select the best motion estimates.

In the next section we propose our Bayesian framework and solution. We will employ the usual model for an observed image M (for mixture) showing a reflection or transparency as a linear combination between the (hidden and underlying) source foreground and background layer images F and B as follows

$$M = F + B \tag{7.1}$$

The complete problem is that of estimating the motions that F and B exhibit as well as estimating F and B themselves, even though we are not necessarily interested in F and B. In this work we use $\mathbf{u}$ to denote the foreground motion and $\mathbf{v}$ to denote the background motion. Results show that our technique outperforms current multiple motion estimators, can handle large image displacements and temporally active motions and is more computationally efficient than current competing multiple motion estimators. Our technique however generates false motion estimates in regions not containing reflection. We therefore show how to reject such erroneous estimates by discarding regions not containing reflections from the final motion estimates. These regions are detected using the FEAPARD-F algorithm that was presented in chapter 6.

7.1 Bayesian Inference for Multiple Motion Estimation

Our goal is to estimate the two motions in the two layers, $[\mathbf{u}, \mathbf{v}]$, given $\mathbf{M}$, a suitable window of observed frames (in our experiments we use 5 frames) including the current one. Clearly this is best achieved by manipulating $p(\mathbf{u}, \mathbf{v}|\mathbf{M})$. However $[\mathbf{u}, \mathbf{v}]$ are related to $\mathbf{M}$ through $\mathbf{L}$ (the hidden foreground and background layers) in a non-trivial fashion. This means that the normal route of marginalization over $\mathbf{L}$ is difficult. In other words to manipulate $p(\mathbf{u}, \mathbf{v}|\mathbf{M})$ we would normally have to perform the marginalization exercise as follows.

$$p(\mathbf{u}, \mathbf{v}, \mathbf{L}|\mathbf{M}) = \int p(\mathbf{u}, \mathbf{v}, \mathbf{L}|\mathbf{M}) d\mathbf{L} \tag{7.2}$$

This would be computationally unattractive. Instead, we choose to take the more pragmatic approach of treating $\mathbf{L}$ as an auxiliary variable in the estimation of $[\mathbf{u}, \mathbf{v}]$. This also allows us to exploit conditional independence in the sense that *given* $\mathbf{L}$, $[\mathbf{u}, \mathbf{v}]$ are independent of $\mathbf{M}$. Hence we manipulate $p(\mathbf{u}, \mathbf{v}, \mathbf{L}|\mathbf{M})$ as follows.

$$p(\mathbf{u}, \mathbf{v}, \mathbf{L}|\mathbf{M}) \propto p(\mathbf{M}|\mathbf{u}, \mathbf{v}, \mathbf{L}) p(\mathbf{u}) p(\mathbf{v}) p(\mathbf{L}) \tag{7.3}$$

Each of the likelihoods and priors are now more readily defined and overall yield a more computationally attractive algorithm. Note again that we are interested in motion and not necessarily in good hidden layer estimation, hence the quality of the estimates for the auxiliary variable $\mathbf{L}$ is less important to us.

7.1.1 Preamble

To give a preamble to our algorithmic strategy, we observe that feature point tracking on image patches showing reflections inevitably results in tracks that follow one or the other layer through the 5 frame window in this case. This means that for each examined site, motion candidates for each layer can be derived from nearby KLT trajectories [85] of length more than four frames. $M^L_{0:4}$ for the examined motion candidate can then be taken as the five image patches centered along the examined trajectory at t=0:4. Using the work of Weiss [94] which was presented

previously in chapter 2, each motion candidate then can be used to generate an estimate for either F or B. This therefore yields candidates for both the auxiliary variables and the motion. Therefore, our strategy is to evaluate each of these candidates according to expression 7.3 and hence generate an optimal estimate for $[\mathbf{u}, \mathbf{v}]$.

7.1.2 Inference for motion

Assume that at each site in the image we can propose candidates for motion and layer images. Furthermore assume that we have already estimated a number of feature point trajectories. Then inference for $[\mathbf{u}, \mathbf{v}]$ over the examined patch can proceed as follows

$$P(\mathbf{u}, \mathbf{v}, F, B|M, \mathcal{T}) = P(M|\mathbf{u}, \mathbf{v}, \mathcal{T}, F, B) \times P(\mathbf{u}|\mathcal{U}, \mathcal{T}, F)P(\mathbf{v}|\mathcal{V}, \mathcal{T}, B)P(F, B) \tag{7.4}$$

Here $\mathcal{T}$ are the estimated feature point trajectories, $P(M|\mathbf{u}, \mathbf{v}, \mathcal{T}, F, B)$ is the likelihood of generating the observed image given the motion and associated layers, $p(\mathbf{u}, \mathbf{v}|\cdot)$ imposes spatial and temporal smoothness on the generated motion and $P(F, B)$ are priors constraining the appearance of the hidden layers. $[\mathcal{U}, \mathcal{V}]$ are local 8 connected motion neighborhoods in the examined block. The next step is to propose suitable expressions for the likelihood and priors. This is discussed in the following sections.

7.1.2.1 Likelihood

The likelihood of observing M given the layers $[F, B]$ is independent of $[\mathbf{u}, \mathbf{v}]$ and the trajectory information $\mathcal{T}$. Hence we express $P(M|\mathbf{u}, \mathbf{v}, \mathcal{T}, F, B)$ using a modified version of Weiss's likelihood term as follows

$$P(M^L_{n:n+4}|L) \propto \exp -\frac{\lambda_w}{5}\left(\sum_{i=n}^{n+4} 1 - \text{SSIM}(L, M^L_i)\right) \tag{7.5}$$

Here λ_w is a weight to configure the importance of the likelihood and L is the intrinsic/reflectance layer (either foreground or background) extracted by processing $M^L_{n:n+4}$ with the Weiss approach [94]. λ_w is fixed to 1 in all experiments. This expression is similar to the likelihood term of Weiss' layer separation algorithm [94] however instead of using gradients we use the structure part of the SSIM [93]. The likelihood therefore measures the error in estimating L (either foreground or background) using Weiss' technique. We use the SSIM here because it is a dimensionless quantity having a range between 0 and 1. Furthermore using the structural component of the SSIM encourages confidence that this likelihood constrains the appearance of the layer in some useful perceptual way independent of illumination. As Weiss' technique estimates layers up to a constant, we turn off the luminance component of the SSIM wherever it is used in this chapter.

7.1.3 Priors

We can factorize the priors on the motion $[\mathbf{u}, \mathbf{v}]$ into a spatial component $P_s(\cdot)$ and a temporal component $P_t(\cdot)$. Hence

$$P(\mathbf{u}|\mathcal{U}, \mathcal{T}, F)P(\mathbf{v}|\mathcal{V}, \mathcal{T}, B) = P_s(\mathbf{u}|\mathcal{U})P_t(\mathbf{u}|\mathcal{T})P_s(\mathbf{v}|\mathcal{V})P_t(\mathbf{v}|\mathcal{T}) \tag{7.6}$$

where $P_s(\cdot)$ and $P_t(\cdot)$ are independent of F and B. The prior $P_s(\cdot)$ encourages spatial smoothness through using a Gibbs distribution as follows

$$P_s(\mathbf{u}|\mathcal{U}) \propto \exp -\lambda_s \left(\sum_{k \in \mathcal{N}} (||\mathbf{u} - \mathbf{u}_k||^2) \right) \tag{7.7}$$

and similarly for $\mathbf{v}$. The neighborhood $\mathcal{N}$ indexes the 8 neighboring blocks around the examined block and λ_s is the spatial smoothness strength set to 0.02 in most experiments. More details about the different values used for λ_s will be discussed in the results section.

To understand the design of the temporal prior, note that each motion trajectory from $\mathcal{T}$ proposes a motion candidate for the underlying layers. The motion candidate is taken as the difference between the point positions of the examined trajectory along the current and next frames. Mean shift clustering is then used to group trajectories with similar motions together. Each cluster now proposes one motion candidate for the underlying layers and is associated with a weight W. This weight is set to the ratio between the number of vectors assigned to the examined cluster and the total number of vectors derived from the examined KLT trajectories $\mathcal{T}$. Hence the temporal prior for the examined motion candidate (say $\mathbf{u}$) is

$$P_t(\mathbf{u}|\mathcal{T}) \propto \exp -\lambda_t (1 - W) \tag{7.8}$$

This therefore biases motion estimates towards more confident trajectories. λ_t is a weight to configure the importance of this term and set to 1 in all experiments.

The priors for the underlying foreground and background layers L contain 2 components. A distribution P_{dl}, encouraging the layers to be **different**, and P_{tl} encouraging the layers to be temporally smooth in time.

$$\begin{aligned} P(F, B) &\propto P_{dl}(F, B)P_{tl}(F)P_{tl}(B) \\ &\propto \exp -\Big[\lambda_{dl} \left(\text{SSIM}(F, B)\right) + \lambda_{tl} \left(1 - \text{SSIM}(F_n, F_{n-1})\right) + \lambda_{tl} \left(1 - \text{SSIM}(B_n, B_{n-1})\right) \Big] \end{aligned} \tag{7.9}$$

Again we use the SSIM as proxy for measuring image similarity (only structural) and the first term encourages the structural similarity between the two images to be low. The second and third terms simply measure the similarity of consecutive image frames in each layer and hence encourage that to be high for good temporal smoothness. Once more $[\lambda_{dl}, \lambda_{tl}]$ are various smoothness weights where λ_{dl} and λ_{dt} are fixed to 10 in all experiments.

7.1.4 Motion Candidates

Each frame is examined in blocks of 50×50 pels. For each block, all KLT trajectories $\mathcal{T}$ [85] within 150 pels from the examined block center and of length more than four frames are selected. Denote the position of the i^{th} trajectory in the examined frame n by $[x_n^i, y_n^i]$, the motion candidate $[s_x, s_y]$ proposed by this trajectory is taken as

$$s_x = x_{n+1}^i - x_n^i \tag{7.10}$$

$$s_y = y_{n+1}^i - y_n^i \tag{7.11}$$

$[s_x, s_y]$ proposes a forward motion candidate for one of the underlying layers. To group together trajectories corresponding to the same layer, all trajectories $\mathcal{T}$ are collected into coherent clusters by mean shift clustering using the corresponding motion candidates $[s_x, s_y]$ as feature vectors. Here mean shift clustering [18] is used with a bandwidth of 2 pels. The four clusters containing most of the examined trajectories are used to generate 4 candidates for $\mathbf{u}$ and $\mathbf{v}$. Each candidate is estimated as the mean motion vector under each cluster.

7.1.5 Layer Candidates

Consider M_{n-n+4}^L to be the five 50×50 image patches centered along frames $n : n+4$. Each one of those images contains a constant layer L undergoing varying illumination through time. L corresponds to one of the underlying layers at the examined block (either the foreground or the background) and it is estimated using Weiss' [94] technique. Recall that the modified layer separation technique of Sarel et al. [74] (see section 2.2.3 for more details) motion compensates the sequence according to one motion, and then the layer extraction step rejects the moving layer in favor of the effectively stationary layer. Therefore, given four motion candidates, we use Sarel et al. approach to generate four layers from each examined block. As each one of those four layers can be assigned to either F or B, there are $2^4 = 16$ possible combinations of $[F, B]$. Each layer combination is then used to correspond to a candidate for the motion pair $[\mathbf{u}, \mathbf{v}]$. Ultimately, the motion pair corresponding to the best estimates of $[F, B]$ are treated as the motions of the observed mixture.

Figure 7.2 gives some idea on how this works in practice. Here a synthetic sequence is created by mixing Lenna with an image of Oranges using the additive mixing model previously introduced (see equation 7.1). Lenna undergoes a motion with $[u_x, u_y] = [5, 0]$ while the Oranges undergo motion with $[v_x, v_y] = [-5, 0]$. The image on the top right of figure 7.2 shows the feature points and associated trajectories. The middle row shows an example of $M_{n:n+4}^L$ in the region bounded by the green square (shown in the first row, left). Here the sequence has been motion compensated using the first motion candidate, which in this case was $[5, 0]$. As can be seen, this has the effect of keeping Lenna stationary, while the Oranges are moving to the left. When Weiss' algorithm is applied to this set of blocks, the extracted image is shown as the leftmost

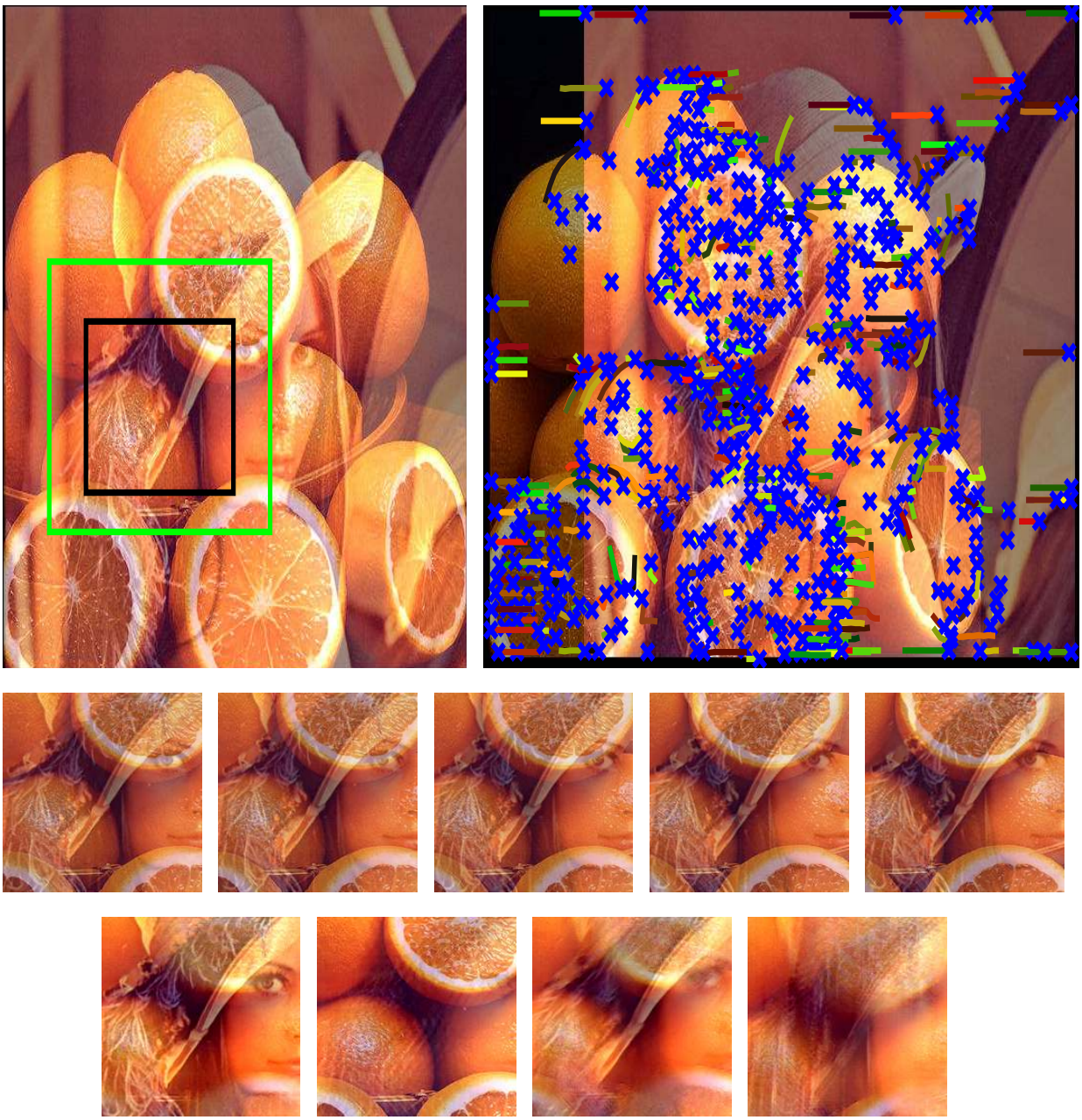


Figure 7.2: *Experiments on a synthetic sequence created by mixing Lenna with Oranges. Top row: First frame from the sequence (left) showing a patch to be considered (shown in green rectangle) and the associated support patch (shown in black rectangle) from which motion candidates (derived from $\mathcal{T}$) are drawn; the corresponding trajectories $\mathcal{T}$ for the 10^{th} frame (right). Each trajectory starts with a blue cross and has a different color. Lenna and Oranges are moving with constant velocities of $[u_x, u_y] = [5, 0]$ and $[v_x, v_y] = [-5, 0]$ respectively. Middle row: $M^L_{n:n+4}$ for the considered patch (shown in green top) using the first motion candidate. Bottom row: Layers extracted with Weiss' technique for each motion candidate. Motion candidates are (from left) $[5, 0]$, $[-5, 0]$, $[3, 3]$ and $[5, -7]$ respectively. As shown, the best layer estimates manifest in the first two images and these correspond to the correct motions of the layers.*

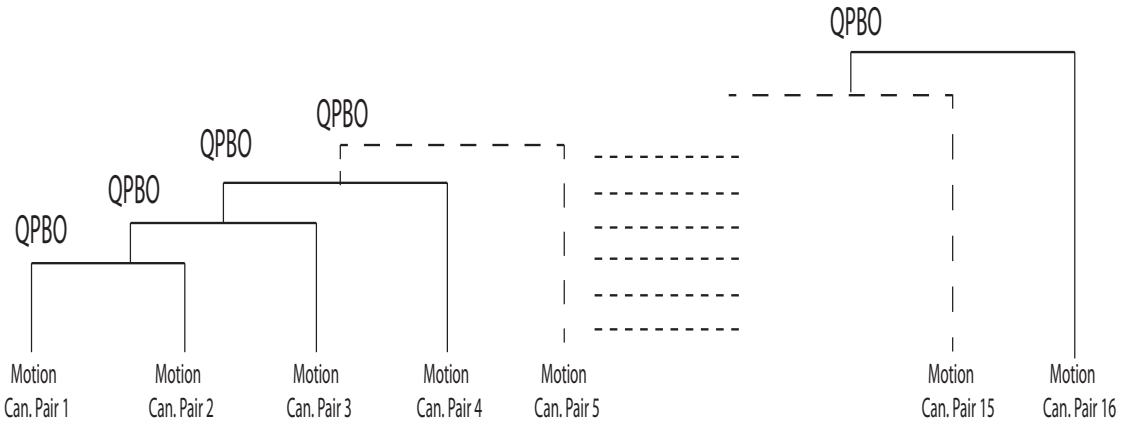


Figure 7.3: *The process of selecting the optimal motion candidate pair from a set of 16 possible candidates. Two candidates are processed at a time via QPBO Graph-Cut and the candidate optimizing the MAP solution is selected.*

image in the last row. As can be seen it corresponds more closely to the image of Lenna than the image of Oranges. The layers resulting from the use of the other 3 motion candidates in this case, are shown as the other images in the bottom row of that figure. The best layer estimates can be visually seen to be the first and second pictures in that row and these do indeed correspond to the correct motions for the examined block.

7.1.6 Final Solution with GraphCuts

Given four candidates for motion and layers above, there are 16 possible $[\mathbf{u}, \mathbf{v}]$ combinations. The MAP estimate for the motions requires maximization of $p(\mathbf{u}, \mathbf{v}, \mathbf{L}|\mathbf{M})$ w.r.t $[\mathbf{u}, \mathbf{v}]$. This is done by choosing between two $[\mathbf{u}, \mathbf{v}]$ candidates at a time using QPBO Graph Cuts [69]. The winning candidate is then processed with the next solution candidate. This process is iterated with the remaining candidates until all 16 candidate pairs are considered (see figure 7.3). All blocks for each frame are processed. To process the first frame at $n = 0$ the temporal part of equation 7.9 is not considered.

Figure 7.4 shows the main components of our multiple motion estimator. We call our technique ***B**ayesian **I**nference for **M**ultiple motion estimation through layer **S**eparation* i.e. **BIMS**.

7.2 Results

7.2.1 Our implementation of competing methods

To compare with previous work, we implemented two versions of competing methods; one relating to optic flow and the other to the Fourier transform. For the optical flow approach we assume motion is constant within a block of 50×50 pels. We then estimate the motion by

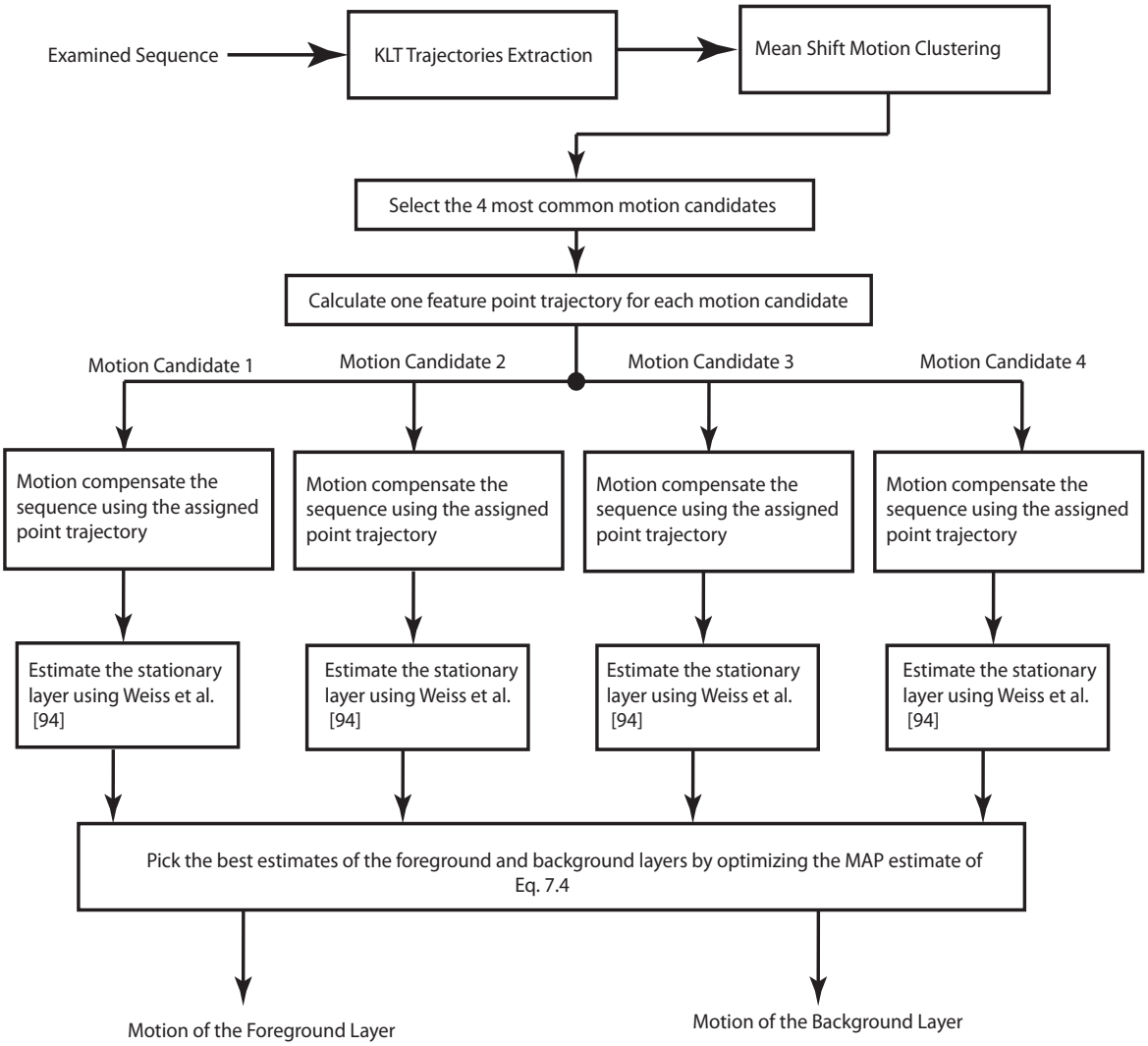


Figure 7.4: *Main Components of the BIMS System.*

solving the OFCE using least square fitting. Note that none of the previous optical flow approaches attempted to solve the motion-layer labeling problem. However in our implementation of the optical flow approach we solve the motion-layer labeling by imposing spatial smoothness on the generated motions as discussed by equation 7.7. Here we have 2 motion candidates per examined block, one for each layer. This generates 4 motion combinations per block. We impose temporal smoothness on the generated motions and we call this technique OPTIC.

For the Transform-Based approach we implement the technique of Stuke et. al [81]. Their technique solves the motion-layer labeling problem using MRFs, imposes temporal smoothness on the generated motions, and uses ICM to maximize the maximum-a-posteriori solution. We

implement this approach using Graph Cuts. For each block, we estimate $[\mathbf{u}, \mathbf{v}]$ using block matching. For each motion component we search displacements -8:2:8 (MATLAB notation). This generates $9^4 = 6561$ possible motion combinations of $[u_x, u_y, v_x, v_y]$ for each block. The 16 motion combinations/pairs optimizing the maximum likelihood solution the most are treated as a pool of motion candidates for the examined block. We call this technique FTRANS. For an examined sequence, the spatial energy weight λ_s (see equation 7.7) have the same value in BIMS, OPTIC and FTRANS.

7.2.2 Experimental Procedures

We test our technique on synthetic and real sequences. In an experiment a synthetic sequence of 53 frames is created by mixing an image of Lenna with an image of Oranges using the additive mixing model (see figure 7.8 first row, left). We call this sequence **LennaOranges**. A repetitive motion model with strong temporal inconsistencies is applied to each layer to examine the robustness of BIMS to camera shake. Lenna is moving with $u_n = [0, 2]$, $u_{n+1} = [-2, -3]$ and $u_{n+2} = [2, 3]$ at $[n, n+1, n+2]$ respectively while Oranges is moving with $v_n = [0, -2]$, $v_{n+1} = [2, 3]$ and $v_{n+2} = [-2, -3]$. This motion pattern repeats itself at $[n+3, n+4, n+5]$ till the end of the sequence.

We also examine our technique on 8 real sequences containing 329 frames of size 576×720. Frames from the examined real sequences are shown in figure 7.5-7.6. All real sequences contain a foreground reflection layer superimposed on a background layer. Sequences are shot with handheld cameras at a distance close to the background layer. As a result in most sequences the background layer is moving with a faster speed than the foreground.

7.2.3 Ground-truth

Motions as estimated by the examined techniques are compared against ground-truth estimates. For real sequences we generate ground-truth estimates manually. We estimate one dominant motion for each layer. For each layer we manually track one strong feature point through time and set the motion of the examined layer to the motion of examined/tracked feature point. For one frame, the error ϵ between the estimated and ground-truth motions is defined as follows.

$$\epsilon = \frac{1}{2}\left(1/N \sum_{i \in \mathcal{R}} \|\mathbf{u}(i) - \tau_{\mathbf{u}}(i)\| + 1/N \sum_{i \in \mathcal{R}} \|\mathbf{v}(i) - \tau_{\mathbf{v}}(i)\|\right) \tag{7.12}$$

Here $[\mathbf{u(i)}, \mathbf{v(i)}]$ are the calculated (2 component) foreground and background motions at site i while $[\tau_{\mathbf{u}}(i), \tau_{\mathbf{v}}(i)]$ are the corresponding ground-truth estimates. $\mathcal{R}$ is the examined region and N is the number of examined sites/blocks in this region. For the sake of accuracy we manually define $\mathcal{R}$ for each frame. As we are just interested in evaluating our technique in regions where single motion estimators fail, we define regions of reflections as ones having two distinctive semi-transparent layers moving over each other.

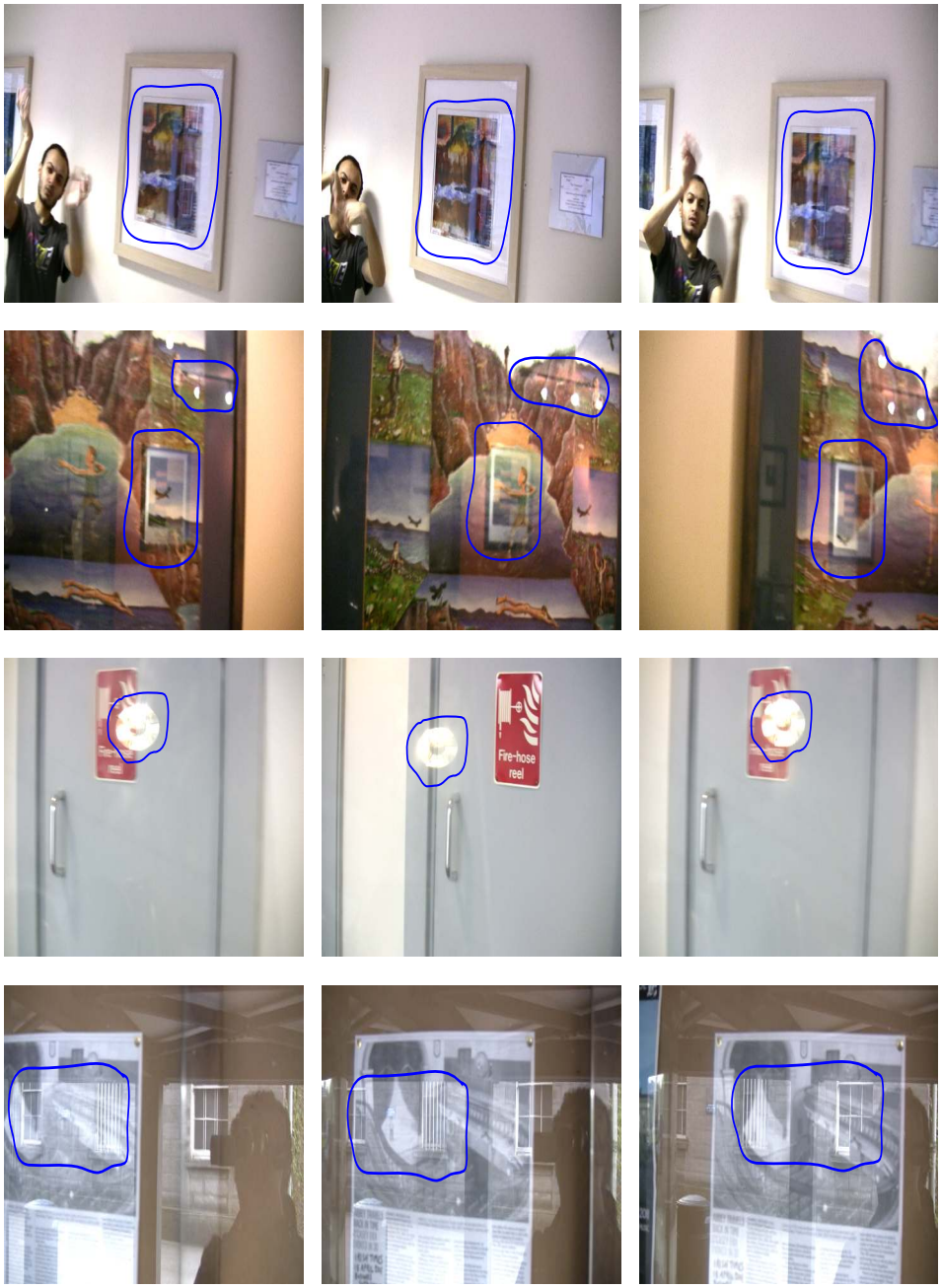


Figure 7.5: *Examined real sequences with regions of interest/reflection $\mathcal{R}$ shown in blue. From top; Frames 5, 15, 25 from* ***SelimK2****; Frames 15, 30, 45 from* ***PicRef****. Frames 5 15, 25 from* ***Bulb****; Frames 10, 30, 45 from* ***WindRefNoShk****.* ***SelimK2*** *contains motion acceleration and* ***PicRef*** *contains camera shake. We generate strong temporal motion discontinuities in* ***Bulb*** *by dropping every third frame from the original sequence.*

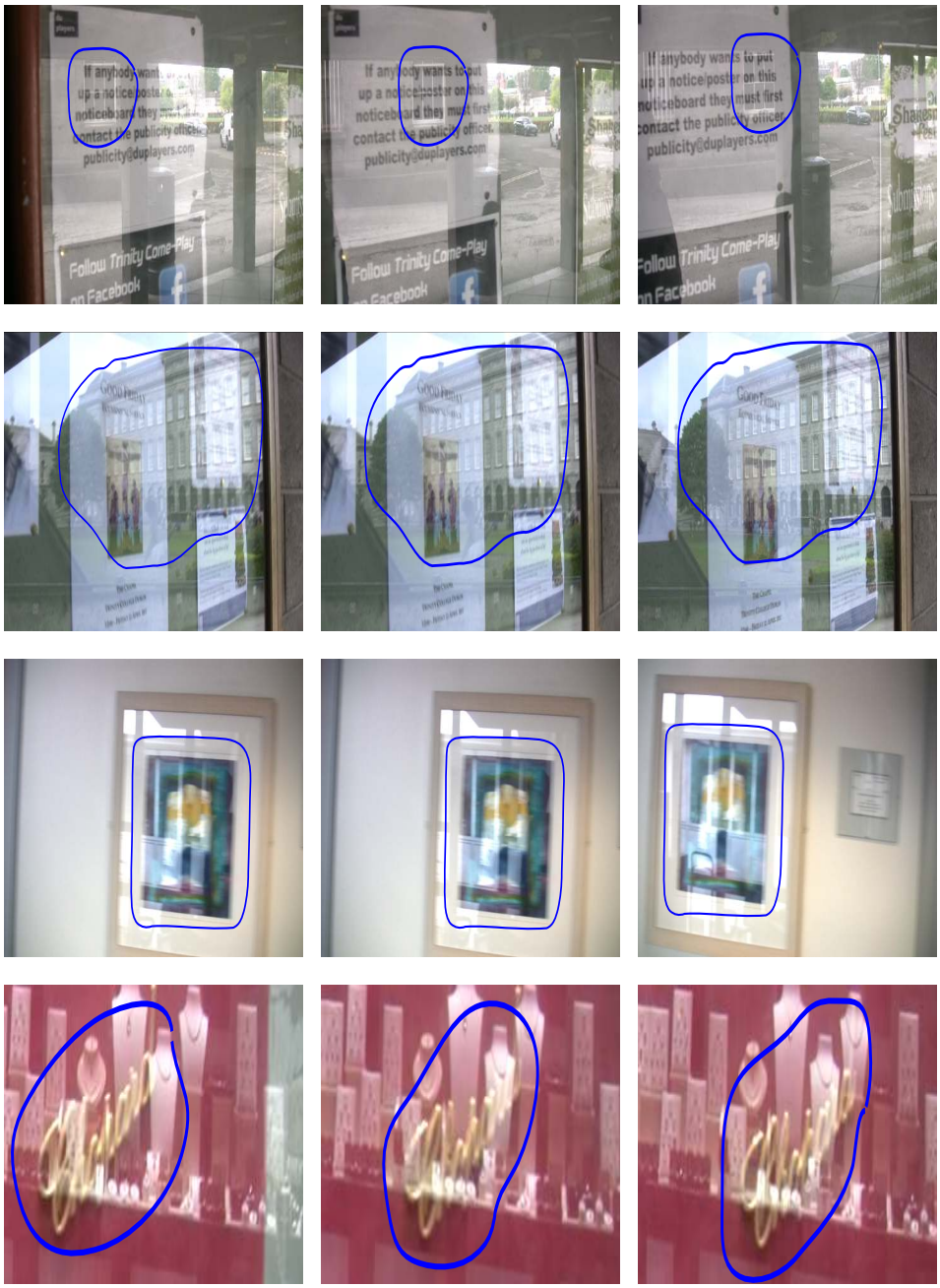


Figure 7.6: *Examined real sequences with regions of interest/reflection $\mathcal{R}$ shown in blue. From top; Frames 5, 15, 25 from* ***WindRefShk****; Frames 5, 10, 15 from* ***WindOnBuild3****; Frames 1, 15, 30 from* ***PortraitA2****; Frames 33, 38, 40 from* ***RedBack****.*

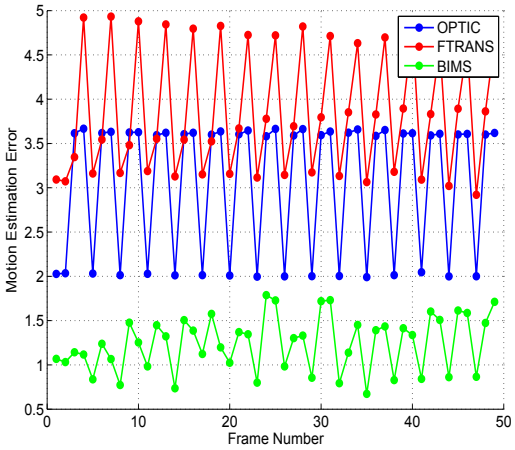


Figure 7.7: *Mean absolute motion estimation error as generated from processing* ***LennaOranges*** *with BIMS, FTRANS and OPTIC. As shown, BIMS outperforms FTRANS and OPTIC.*

7.2.4 Synthetic Data

Figure 7.7 shows the motion estimation errors generated from processing **LennaOranges** with BIMS, OPTIC and FTRANS. Here $\lambda_s = 0.5$. As shown, OPTIC and FTRANS generated large motion errors as they assume constant motion over three frames, an assumption that is violated in the examined sequence. However, our technique was able to handle this motion behavior as BIMS does not assume constant motion over any frame window. In addition, unlike previous techniques such as Stuke et al. [81], BIMS imposes temporal smoothness on the separated layers not on the generated motions. This generated motion-layer labeling that is temporally consistent.

Figure 7.8 shows the estimated motions for frame 11 (left column) and 12 (right column) for the synthetic sequence using (from top) BIMS, FTRANS and OPTIC. Groundtruth motions are $u_n = [-2, -3]$ and $u_{n+1} = [2, 3]$ for Lenna and $v_n = [2, 3]$, and $v_{n+1} = [-2, -3]$ for Oranges. As shown, the motion-layer labeling generated by BIMS (first row) is temporally and spatially consistent. However, both FTRANS (second row) and OPTIC (last row) generated large motion errors as the examined motions undergo strong temporal motion discontinuities. In addition, OPTIC generated small motion estimates. Such behavior is expected for optical flow approaches as the OFCE is only valid given small image displacements. Single motion estimators overcame this assumption by iteratively refining motion estimates and shifting the examined frame with the updated motions [48]. However none of the optical flow multiple motion estimators proposed to iteratively refine motion estimates. One should note that for reflections this is not an easy problem as it requires shifting each layer with its estimated motion. This will require an intermediate stage that performs layer separation, a problem that requires motions to be known beforehand.

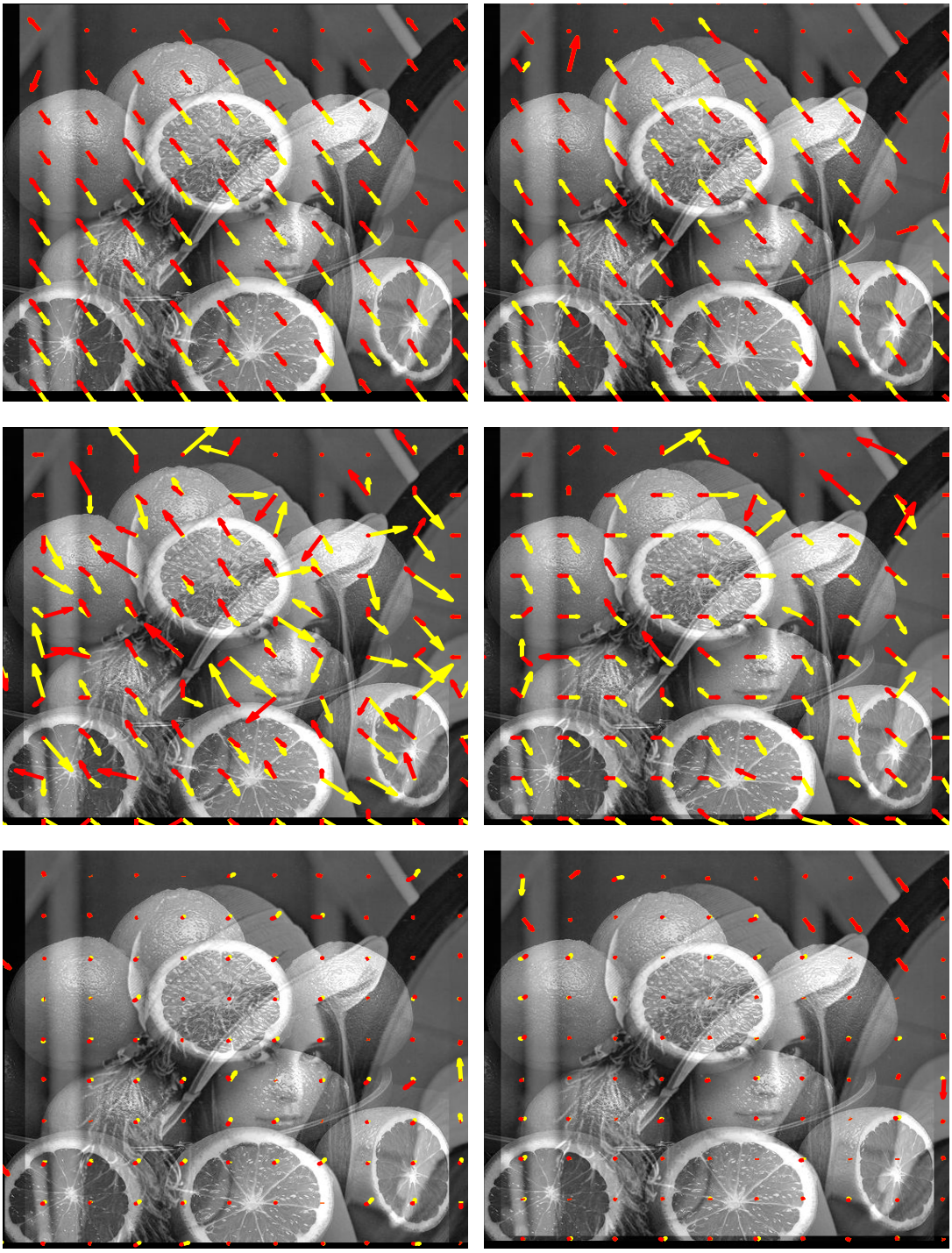


Figure 7.8: *Multiple motions for frame 11 (left column) and frame 12 (right column) as estimated by (from top), BIMS, FTRANS and OPTIC. Red and yellow vectors represent the estimated motions of Lenna (foreground) and Oranges (background) respectively. Here we scale the estimated motions by a factor of* 5 *and show the grayscale representation of the image mixtures for illustration clarity. BIMS motion estimates for the two frames are* $u_n = [-2, -3]$, $v_n = [2, 3]$, $u_{n+1} = [2, 3]$ *and* $v_{n+1} = [-2, -3]$ *respectively. These match the ground-truth estimates. In addition, BIMS generated motion-layer labeling that is temporally consistent despite the strong temporal motion discontinuities in the examined sequence. However, FTRANS and OPTIC failed to handle the strong temporal motion discontinuities. BIMS usually outperforms FTRANS and OPTIC given good enough KLT tracks are available.*

7.2.4.1 BIMS Parameters Configuration

To illustrate the importance of the likelihood in equation 7.5 and the priors in equation 7.7-7.9, we change the values of their weights (denoted by λ) and reprocess frame 12 of **LennaOranges**. Figure 7.9 (first row, left column) shows the result of switching OFF the temporal consistency term of equation 7.9 by setting $\lambda_{tl} = 0$. This generated motion-layer labeling that is temporally inconsistent with frame 11 (in figure 7.8, first row, left). More explicitly, the background motion at frame 11 (shown in yellow) got assigned to the foreground layer in frame 12 (shown in yellow). Similarly, the foreground motion at frame 11 (shown in red) got assigned to the background layer in frame 12 (shown in red). To illustrate the importance of the layer structural independence term, we switch OFF its term by setting $\lambda_{dl} = 0$. The result is that the same vector got assigned to both layers as shown in figure 7.9 (first row, right column). The reason for this result is that equation 7.5 now assigns the same separated layer to both F and B. Last, figure 7.9 (last row) shows the result of switching OFF the motion spatial smoothness term by setting $\lambda_s = 0$. This generates motion-layer labeling that is spatially inconsistent as shown in the purple boxes.

7.2.5 Real Data

Figure 7.10-7.11 shows the motion estimation errors generated from processing eight real sequences with BIMS, FTRANS and OPTIC. Image sequence results can be found in the accompanying DVD. Table 7.1 summarizes the motion errors. As shown our technique outperforms FTRANS and OPTIC in all sequences and in most of the examined frames.

Figure 7.12-7.14 shows motion results on some frames from the examined sequences. Here we show the grayscale representation of the examined sequences for illustration clarity. BIMS handled well the camera shake/motion acceleration in **SelimK2**, **PicRef**, **Bulb** and **WindRef-Shk** (see figure 7.12-7.13). FTRANS and OPTIC however generated large motion errors in those sequences (see figures and table 7.1). In all sequences BIMS generates smoother results than FTRANS and OPTIC. In **PortraitA2** (see figure 7.14, left) FTRANS generates large motion estimation errors especially for the purple region. In this region the vertical white bar is moving to the left and hence generates an aperture effect. FTRANS failed to handle this effect as it does not incorporate enough global information. However BIMS handled this effect successfully as KLT trajectories near this region capture global motion information that are good enough to recover the motion of the white vertical bar. More results can be found in Appendix D.

7.2.6 Reflection Detection for Multiple Motion Estimation

Processing an entire sequence with BIMS may generate false estimations in regions where there is no reflection. Figure 7.12 (first row, shown in purple) shows an example of this scenario. Here the actor is moving his hands quickly generating pathological motion. Processing the actor's hands with BIMS generates erroneous measurements. Hence it is required to mask out this region from the BIMS motion estimates. An approach to do so is by manually selecting

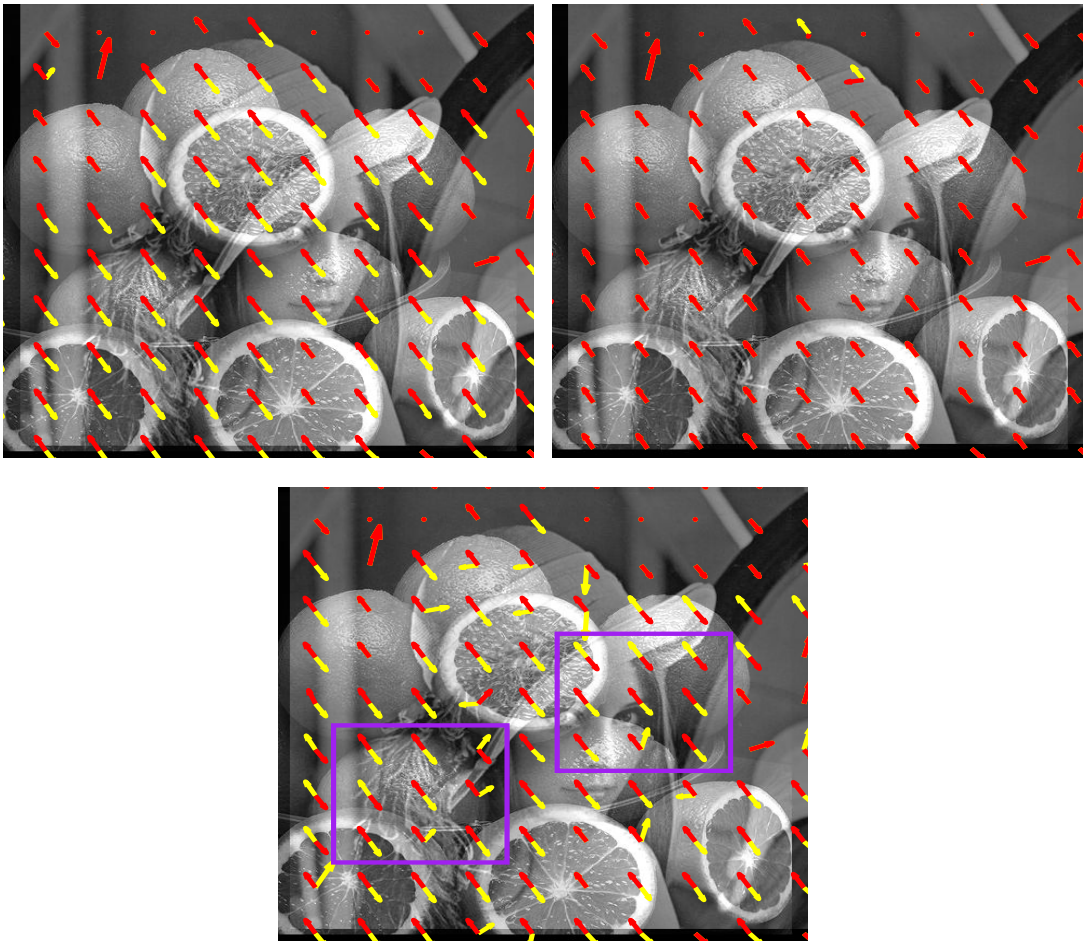
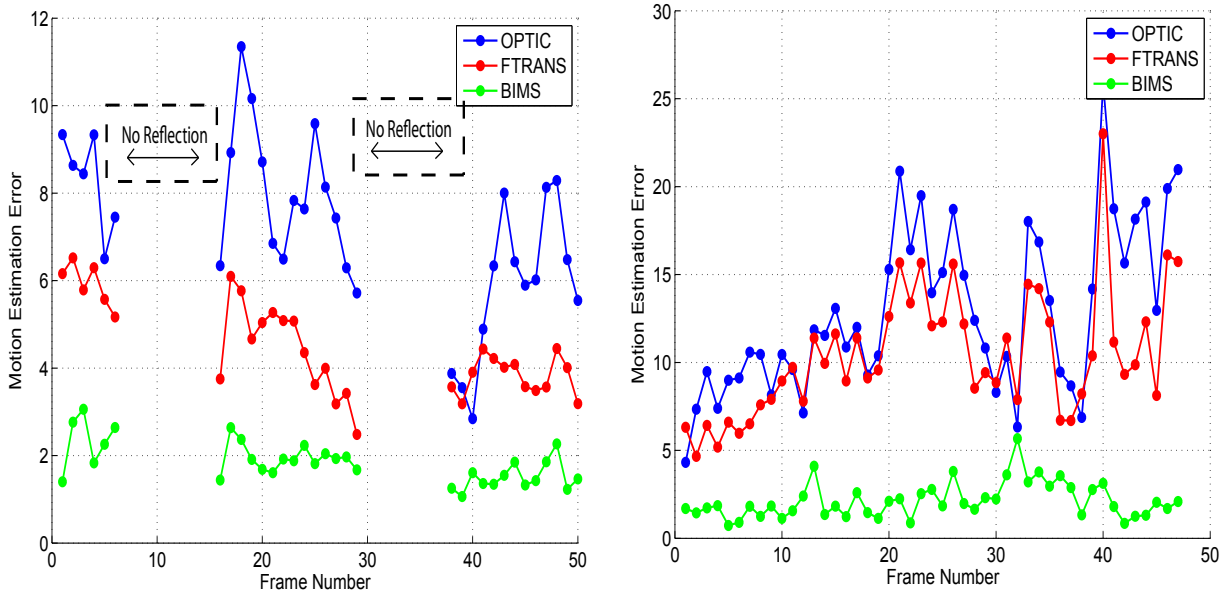
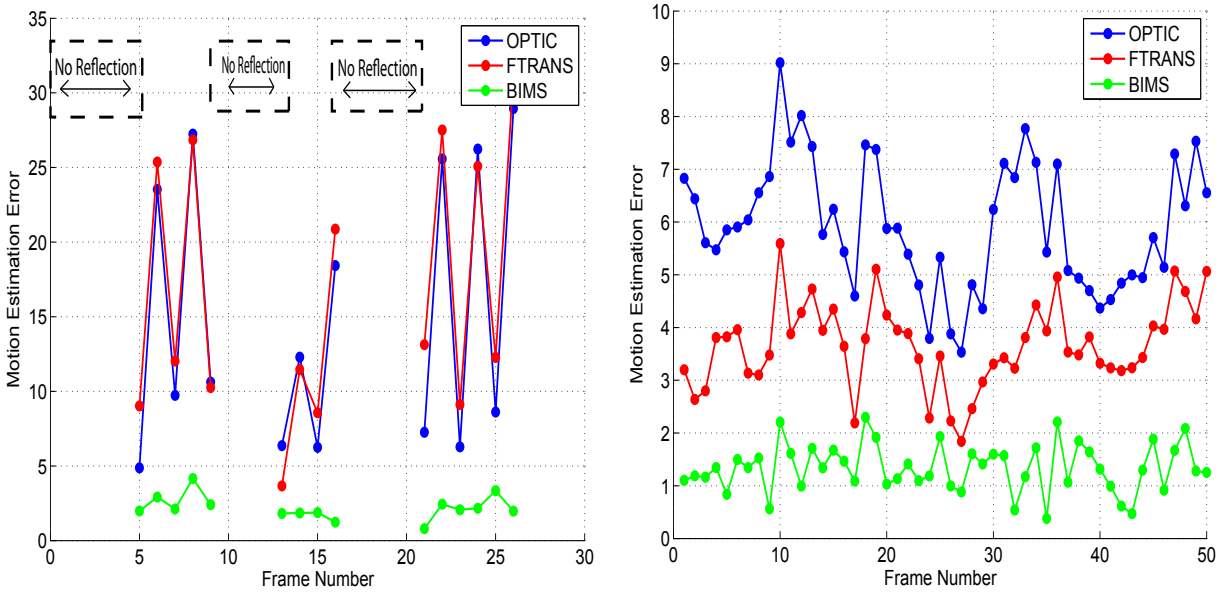


Figure 7.9: *Examining the effect of the likelihood and priors (of equation 7.5 and equation 7.7-7.9) by processing frame 12 of* ***LennaOranges****. Top row, left: Turning OFF the layer temporal consistency term from BIMS generates motion-layer labeling that is temporally inconsistent with motion estimates of frame 11 (see figure 7.8 first row, left); Top row, right: Turning OFF the layer structural independence term from BIMS assigns the same motion vector to both layers; Last row: Turning OFF the motion spatial smoothness term from BIMS generates motion-layer labeling that is spatially inconsistent (see the purple boxes)*

the regions of interest/reflections $\mathcal{R}$ as shown in blue in figure 7.12. This however requires large manual intervention. Instead we use our reflection detection technique FEAPARD-F (see Chapter 6) to estimate the regions of interest. Figure 7.15-7.16 shows reflection masks for four real sequences generated by our FEAPARD-F. Those masks are used to weight the motion estimates in regions not containing reflection. In figure 7.15 (first column) the pathological motion of the actor's hands is successfully discarded from the final motion estimates. In the examined four sequences FEAPARD-F was able to discard regions not containing reflections from the final BIMS estimates.

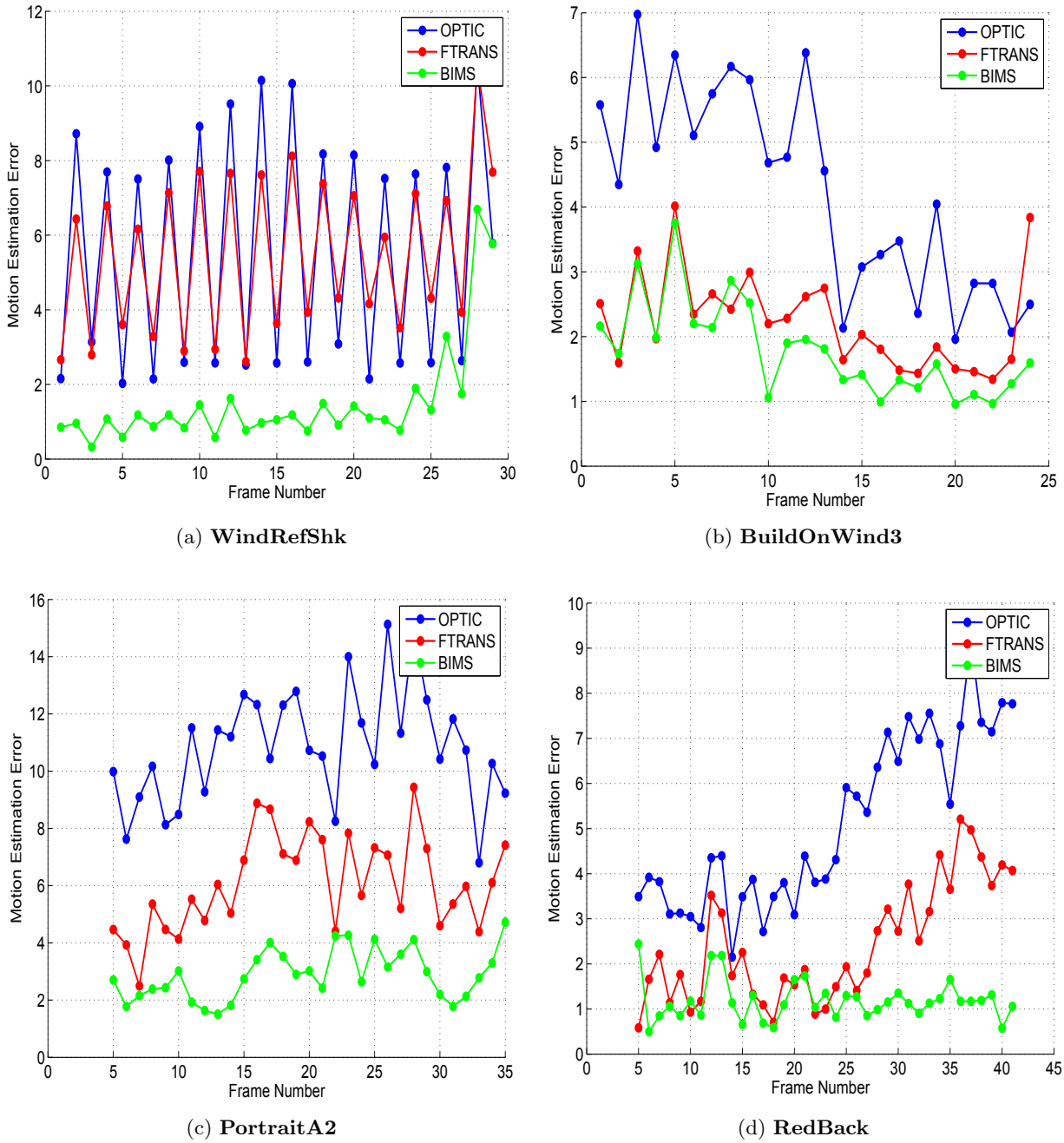


(a) **SelimK2** (b) **PicRef**



(c) **Bulb** (d) **WindRefNoShk**

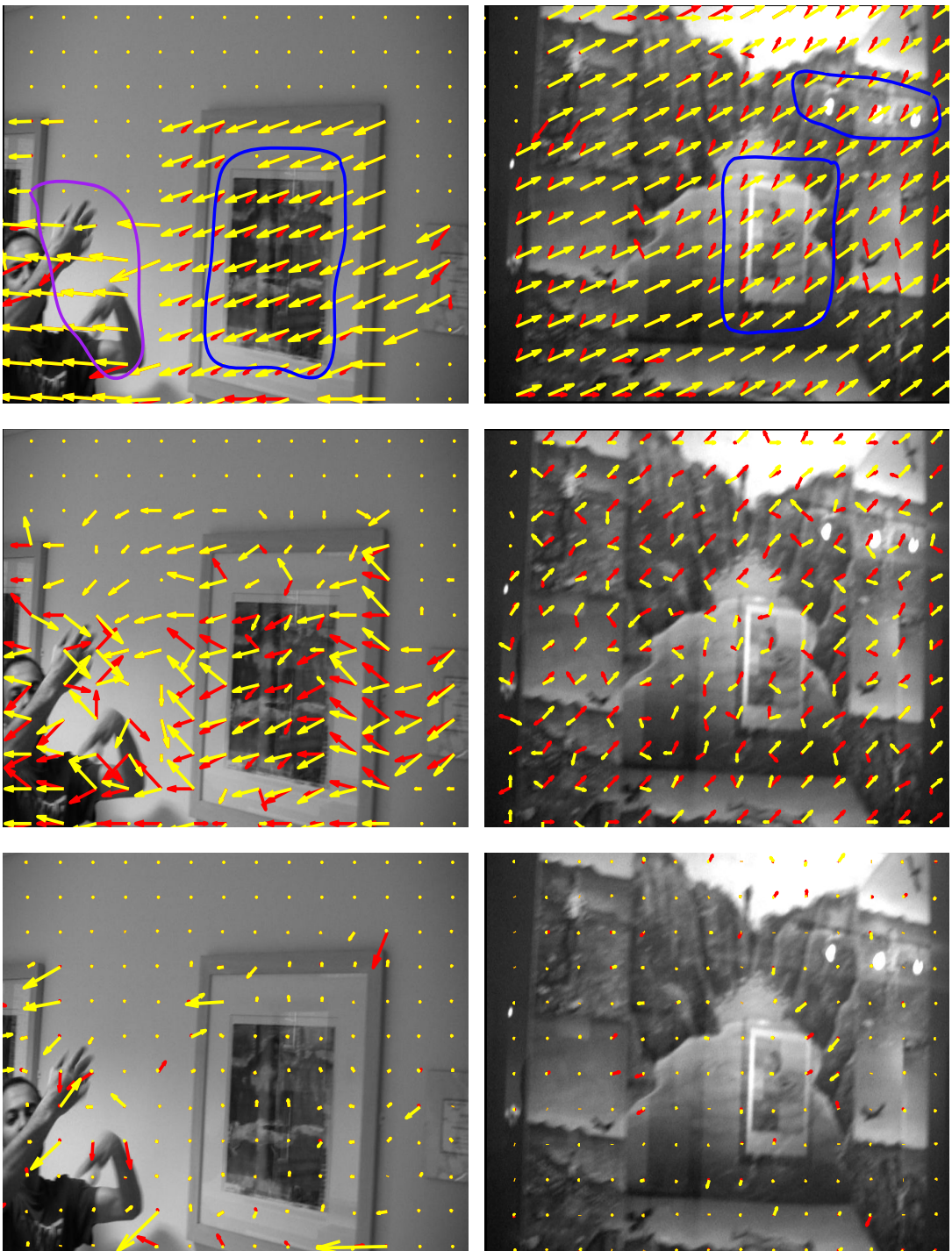
Figure 7.10: *Mean absolute motion estimation errors generated from processing four real sequences with different motion estimators. λ_s is set here to (in clockwise direction);* 0.05, 0.01, 0.2 *and* 0.01 *respectively. Some frames in* ***SelimK2*** *and* ***Bulb*** *do not have reflections and hence they do not have motion estimates (see the black dashed boxes). Our technique BIMS outperforms FTRANS and OPTIC in all sequences.*



(a) **WindRefShk** (b) **BuildOnWind3**

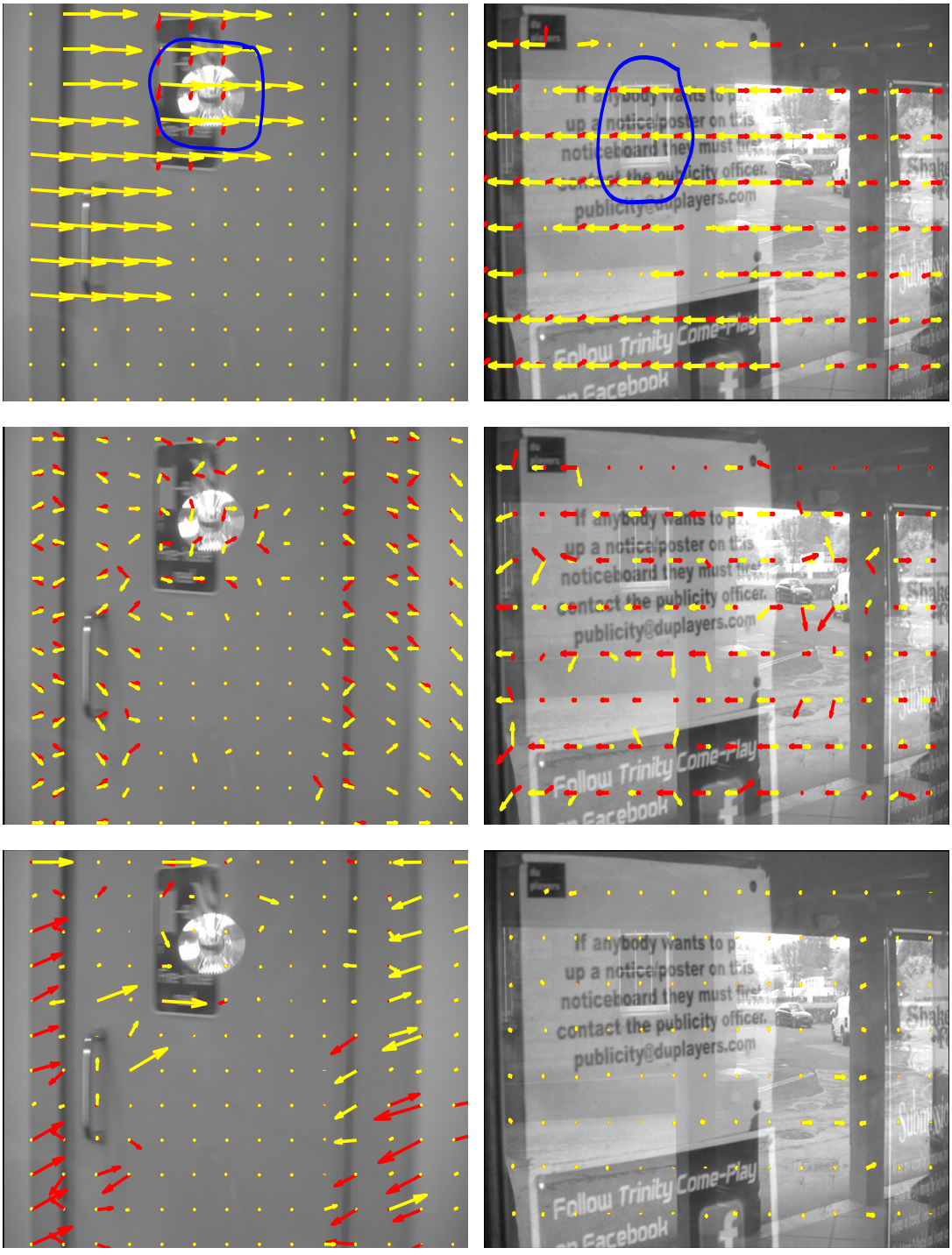
(c) **PortraitA2** (d) **RedBack**

Figure 7.11: *Mean absolute motion estimation errors generated from processing real sequences with different motion estimators.* λ_s *is set here to (in clockwise direction);* 0.02, 0.02, 0, 05 *and* 0.05 *respectively. Our technique BIMS outperforms FTRANS and OPTIC in all sequences.*



(e) **SelimK2**, frame 44 (f) **PicRef**, frame 27

Figure 7.12: *Motion estimation results for two sequences as generated by (from top); BIMS, FTRANS and OPTIC respectively. Background and foreground motion are shown in yellow and red respectively. Motions are scaled by a factor of 5 (left) and 2 (right) for illustration clarity. Regions of interest $\mathcal{R}$ are shown in blue. BIMS was able to handle motion acceleration and camera shake in both sequences. However it generated large errors in regions of pathological motion (shown in purple). We discard this region from the final motion estimation using our reflection detection technique (see section 7.2.6).*



(e) **Bulb**, frame 6 (f) **WindRefNoShk**, frame 32

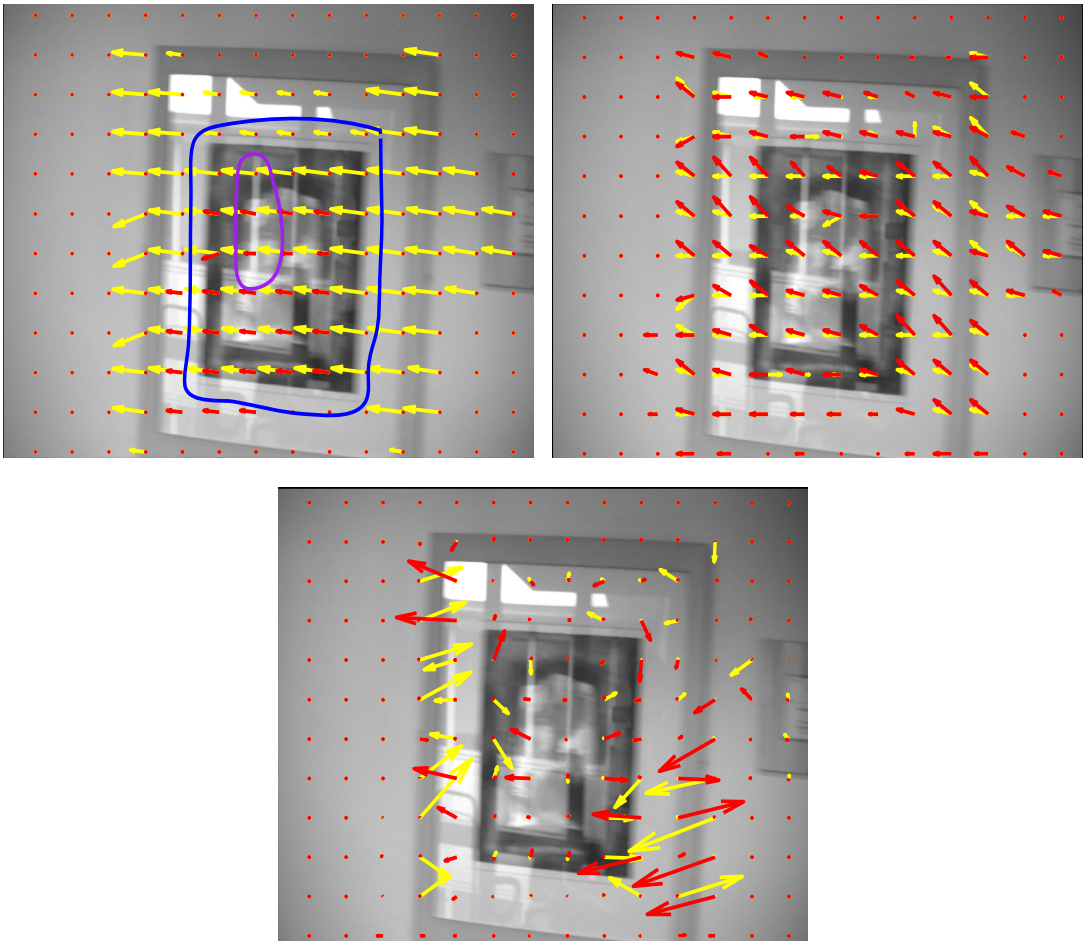
Figure 7.13: *Motion estimation results for two sequences as generated by (from top); BIMS, FTRANS and OPTIC respectively. Background and foreground motions are shown in yellow and red respectively. Motions are scaled by a factor of 2 (left) and 3 (right) for illustration clarity. Regions of interest $\mathcal{R}$ are shown in blue. BIMS was able to handle the strong camera shake in* ***Bulb*** *however FTRANS and OPTIC generated large errors in this sequence. BIMS generated smoother results in* ***WindRefShk*** *than the other techniques.*

	BIMS	FTRANS	OPTIC
SelimK2	1.8±0.5 1.1 3.1	4.5±1.1 2.5 6.5	7.2±1.9 2.8 11.3
Bulb	2.2±0.8 0.8 4.2	16.4±8.7 3.7 31.1	14.8±9.1 4.9 28.9
PicRef	2.15±1 0.7 5.7	10.4±3.6 4.7 23.0	12.8±4.7 4.3 25.8
WindRefNoShk	1.3±0.4 0.4 2.3	3.7±0.8 1.8 5.6	5.9±1.2 3.5 9.0
WindRefShk	1.4±0.3 0.32 6.7	5.5±2.1 2.6 10.6	5.2±3.1 2.0 10.6
BuildOnWind3	1.8±0.7 1.0 3.8	2.2±0.7 1.3 4.0	4.3±1.6 2.0 7.0
PortraitA2	2.8±0.8 1.5 4.3	6.1±1.7 2.5 9.4	5.4±1.5 2.5 8.1
RedBack	1.2±0.4 0.5 2.4	2.4±1.2 0.6 5.2	5.1±1.8 2.1 9.2

Table 7.1: *Mean absolute motion estimation errors for OPTIC, FTRANS and BIMS on* 8 *real sequences. For each technique we show the mean, minimum and maximum error over the frames of the examined sequence. As shown our technique BIMS generates the least error in all examined sequences.*

7.2.7 Layer Separation

Figure 7.17 shows layer separation examples generated from BIMS. Figure 7.17 shows that BIMS reduces reflections in the extracted layers. In figure 7.17 (first row), the foreground layer (white bulb) is well separated from the red background. In figure 7.17 (second row), the white foreground layer is well separated from the (background) yellow writing. Please note that for motion estimation we are not interested in perfect layer separation, instead we are just interested in separation that is good enough for motion estimation. However improving layer separation



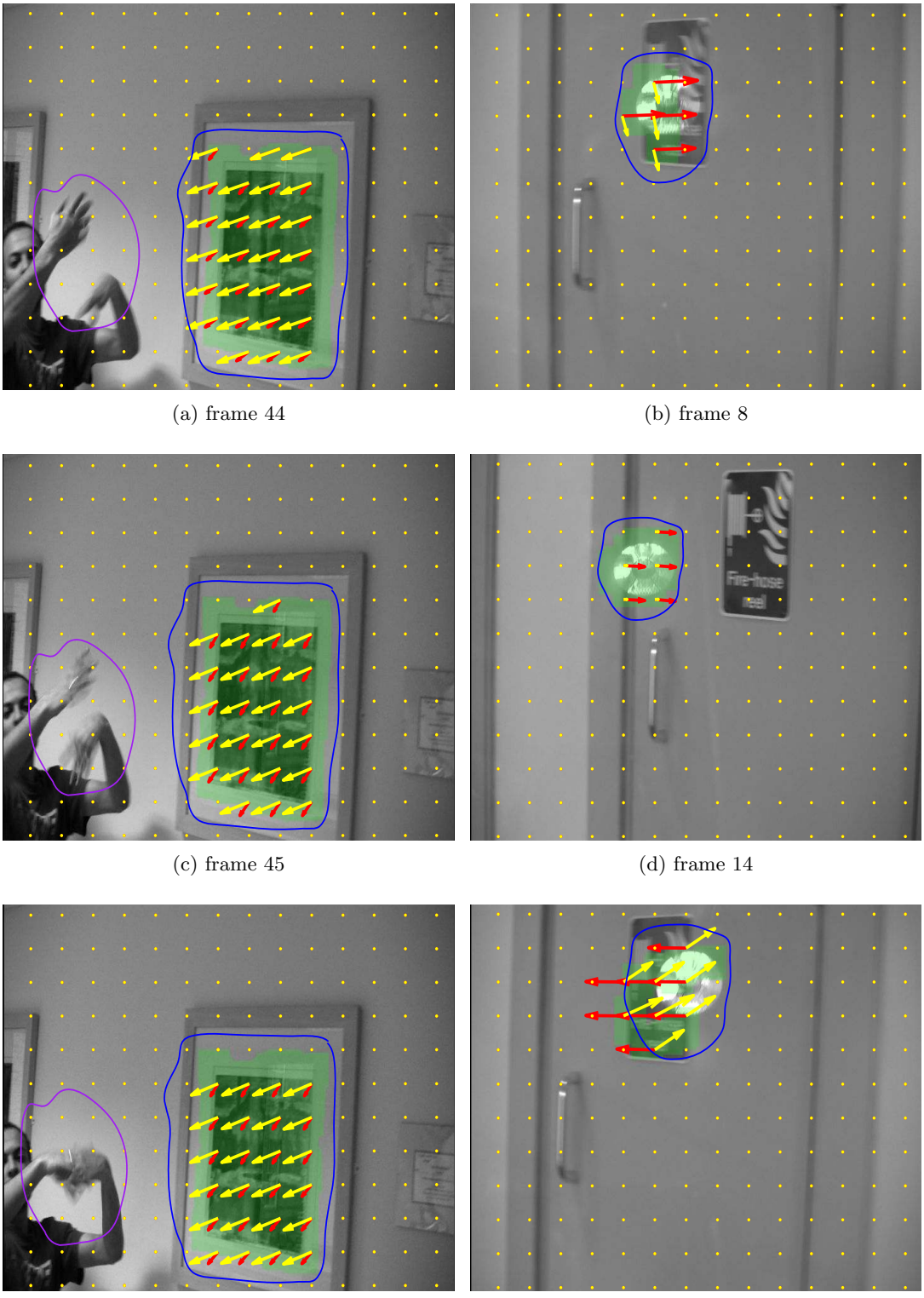
(c) **PortraitA2**, frame 19

Figure 7.14: *Motion estimation results of **PortraitA2** as generated by (in clockwise direction); BIMS, FTRANS and OPTIC respectively. Background and foreground motions are shown in yellow and red respectively. Motions are scaled by a factor of 3 for illustration clarity. Regions of interest $\mathcal{R}$ are shown in blue. BIMS handled the aperture effect well (shown in purple). FTRANS and OPTIC failed to handle this effect as they do not include enough global information.*

will improve motion estimation. This is left for future research.

7.2.8 Computational Complexity

Our technique consists of three main stages. KLT trajectories extraction, Layer Separation using Weiss [94] and MAP optimization using QPBO Graph-cuts. A C++ implementation of KLT trajectories extraction is used [1]. This takes $\approx$ 1 second to process one frame of size 576×720 pels. A C++ implementation of QPBO Graph-Cuts is also available. This stage is computationally efficient as it is applied on a block basis not on pel-basis. For a frame in standard definition and blocks of size 50×50 pels, there are 11×14 blocks/sites examined

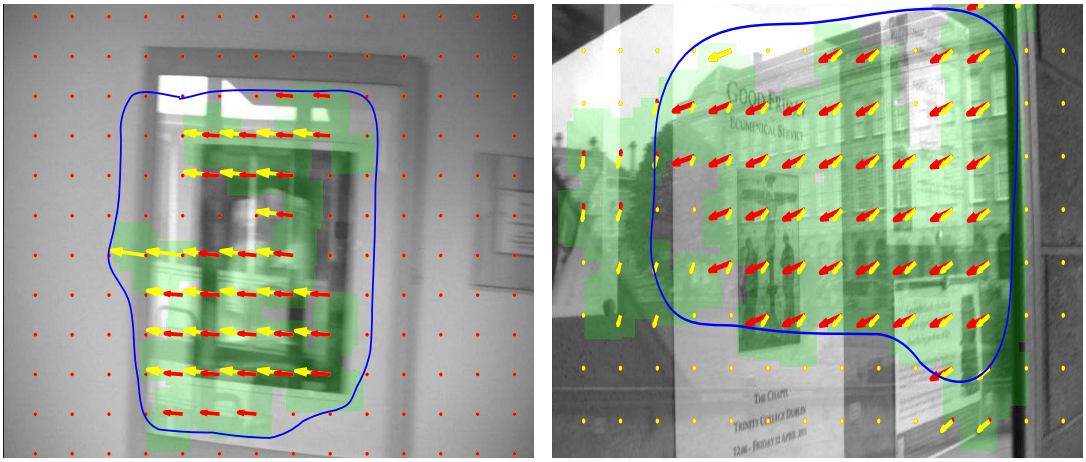


(a) frame 44 (b) frame 8

(c) frame 45 (d) frame 14

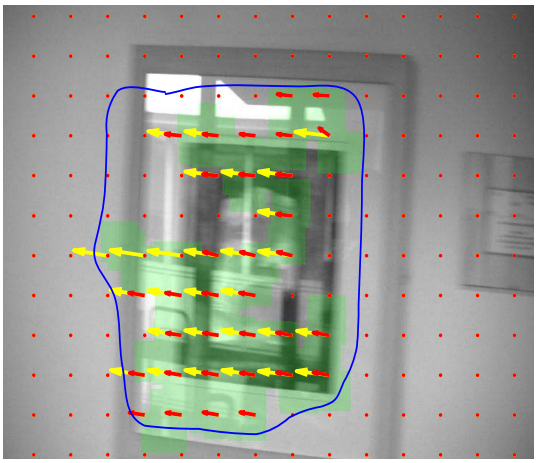
(e) **SelimK2**, frame 46 (f) ***Bulb***, *frame 26*

Figure 7.15: *Motion estimation of BIMS with automated generated reflection detection masks (shown in green). Yellow and red vectors show the background and foreground motions respectively. Reflection detection masks are generated using our reflection detection technique FEAPARD-F (for more details see Chapter 6). Detection masks are used to weight out erroneous measurements in regions not containing reflections. Ground-truth region of reflections are shown in blue. FEAPARD-F was able to discard the pathological motion of actor's hands (shown in purple) from the final BIMS estimates.*

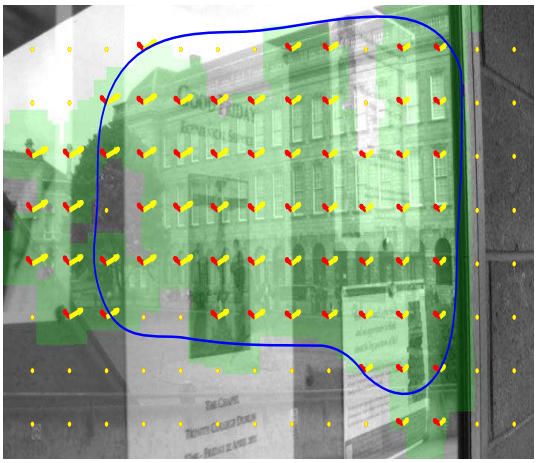


(a) frame 20

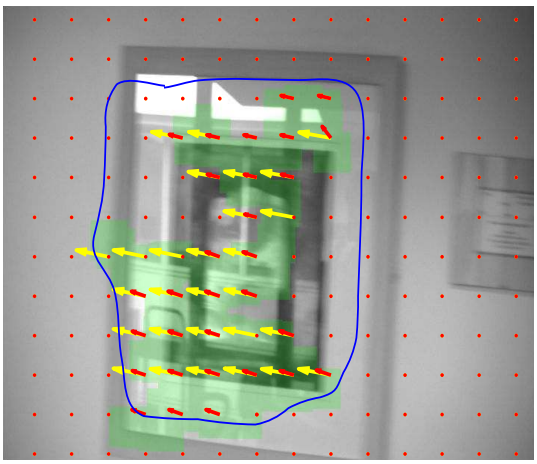
(b) frame 12



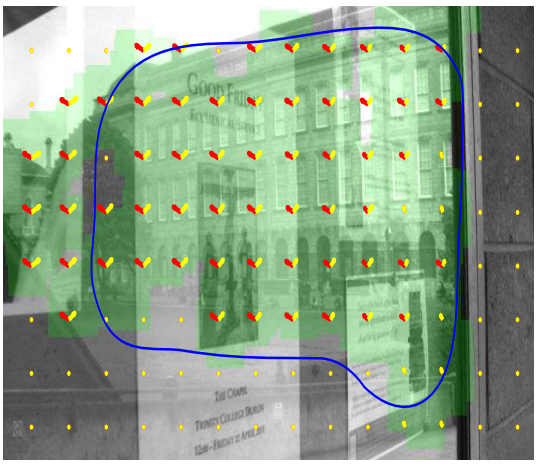
(c) frame 21



(d) frame 15



(e) **PortraitA2**, frame 22



(f) **BuildOnWind3**, frame 22

Figure 7.16: *Motion estimation of BIMS with automated generated reflection detection masks (shown in green). Yellow and red vectors show the background and foreground motions respectively. Reflection detection masks are generated using our reflection detection technique FEAPARD-F (for more details see Chapter 6). Detection masks are used to weight out erroneous measurements in regions not containing reflections. Ground-truth region of reflections are shown in blue. FEAPARD-F was able to discard the regions not containing reflections from the final BIMS estimates.*

Figure 7.17: *Left column, from top; Frames from* ***Bulb***, ***RedBack*** *and* ***PicRef****. Last two columns; Extracted foreground (middle) and background (last) layers using BIMS. As shown BIMS reduces reflection in the extracted layers.*

by QPBO. The most computationally expensive stage in BIMS is layer extraction using Weiss approach. To calculate the layer for one motion candidate, it requires 4×5 spatial derivatives, 2×5 fast fourier transforms operators and 2×5 inverse fast fourier transforms operators. For one examined site and 4 motion candidates, the Weiss approach requires 80 spatial derivatives, 40 FFTs and 40 IFFTs.

Optical flow approaches are computationally fast as closed-form solutions exist. However they generate poor motion estimates. Transform-Based approaches mainly use block matching to solve for motions. For an examined site, K^4 block matching operations are required if one searches K possible displacements for each motion component. Hence there are 6561 and 83521 block matching operations for $K = 9$ and $K = 17$ respectively. Note in this technique there is an explosion in computational complexity in return for motion estimation accuracy. Searching a

small range of motions reduces computational load but increases the chance of motion estimation errors. This problem does not exist in BIMS as motion candidates are proposed by the KLT trajectories.

The average time for processing one standard definition frame with BIMS and FTRANS is ≈ 1.3 and ≈ 3.2 minutes respectively. Here we use $K = 9$ for FTRANS and the average time is taken over a sequence of 50 frames. Both approaches are implemented using MATLAB and on the same processor. By setting $K = 17$ the average processing time for one frame with FTRANS is ≈ 28 minutes. Note the explosion in computational load.

7.3 Conclusion

We have presented a Bayesian framework for multiple motion estimation in regions of reflection. By articulating the problem using layers as auxiliary variables we are able to impose new constraints on the estimated motions through the layers that have been extracted. A key advance is to encourage motions to be estimated in a way which improves the plausibility of the extracted layers. This allows our technique to be robust to temporal motion activity and to give improved motion fields as compared to existing approaches. We also proposed a novel motion candidate selection scheme based on KLT trajectories which allows the joint problem to admit a tractable solution. This makes our multiple motion estimator computationally more efficient than current competing techniques. We showed examples where automated reflection detection can be used to weight out erroneous motion estimates in regions not containing reflection. However, our motion estimator does require distinctive foreground and background layers. The development of an interactive scheme to circumvent this issue would help the tool to be useful in the post-production industry.

8 Conclusion

This thesis focused on handling transparency on digital image sequences. It has three main contributions. The first is a technique for blotch and line scratch removal from image sequences by modeling these degradations a a semi-transparent layer superimposed on the original clean layer. The second contribution is a technique for detecting reflections in image sequences and the last contribution is a technique for multiple motion estimation in regions of reflections. The following sections give a brief review of the work presented in this thesis and propose some ideas for future work in these areas.

8.1 Blotch and Line Removal

Chapter 3 presented an overview of current blotch and line removal techniques. Most of the existing techniques model corruptions as an opaque layer superimposed on the original data. This often leads to overestimation of the corruption borders which introduces more restoration artifacts in clean regions than being removed. The algorithms of JONDI and JOMBEI of Kokaram et al. [46, 47] are the most general framework for blotch and line removal using an opaque corruption model. They are the basis of many commercial software packages.

Few authors have attempted to reduce restoration artifacts in clean regions by using a semi-transparent corruption model instead of an opaque model [14, 20, 32, 79]. Crawford et al. [20] modeled the corrupted data as a linear mixture between the original data and dirt/corruption layer. They achieved removal by estimating the corruption opacities and original data using Bayesian Matting. Bayesian Matting was discussed in detail in Chapter 2. Crawford et al.

however did not work on video and did not take full advantage of the RGB color channels. Instead their matting solution is formulated on the grayscale channel only. In addition they use an inpainting technique with an opaque corruption model to restore the chroma channels. As a result their technique does not handle textured regions well. Last, their technique is not initialized with an automated step for detection.

In Chapter 4 we presented a new technique for blotch removal with an aim to reduce restoration artifacts generated in clean regions during the removal step. Similarly to Crawford et al. we model the corruption as a semi-transparent layer superimposed on the original data. Removal is also achieved using Bayesian Matting. The main difference with Crawford's technique however is that we use temporal priors and we formulate the matting solution on the three RGB color channels. This proved to give better handling of highly textured regions over Crawford's technique. We then extended our blotch removal technique to line scratch removal. Here we modified the temporal priors to cope with the temporally consistent nature of lines. In both blotch and line removal techniques we fuse temporal and spatial priors in a way to maintain reconstruction integrity despite motion and texture complexity. We initialized our blotch remover with an automated step for detection and results show that our technique is capable of discarding a high false detection rate from the removal step.

In Chapter 5 we compared our removal techniques against JONDI and JOMBEI and other current removal techniques. We generated ground-truth estimates using IR scans of corruptions. Here we derived a relation that estimates the corruption opacities from IR scans. We then used the estimated opacities to weight out the blotches/lines from the corrupted sequences. Results show that our removal techniques outperform the existing techniques.

8.2 Reflection Detection in Image Sequences

Chapter 6 presented a technique for detecting regions of reflection in image sequences. Regions of reflections contain two semi-transparent layers moving over each other. As a result they cause most of the current image processing techniques to fail as they assume the presence of one layer per pel i.e. motion estimation, object recognition. We presented a technique for detecting reflections that can be used as a preprocessing step in many video applications. The idea here is to detect regions of failure and then the user can perform a special treatment for the detected regions depending on the application.

As reflections can result by mixing any two layers they come in many shapes and colors. This makes their detection a hard problem that was not addressed before. We detect reflections by analyzing KLT feature point trajectories. Three main characteristics are analyzed 1) layer separability, 2) temporal behavior and 3) image sharpness. We define reflections as regions with low image sharpness and bad temporal match of feature points. In addition, we define reflections as regions that can be decomposed into two distinctive foreground and background layers. We proposed a technique that performs layer separation on a single still image. Even though we are

not interested in the quality of separation itself, results show that the separated layers provide valuable information for reflection detection.

We presented 9 weak detectors analyzing the three main features of reflections. We then generated a strong reflection detector by combining the 9 weak detectors. We imposed spatial and temporal smoothness on the detection fields and we proposed an automated approach for system parameter configuration. Results show detection of several forms of reflections with rejection to pathological motion i.e. motion blur, occlusion.

8.3 Multiple Motion Estimation for Regions of Reflections

Chapter 7 presented a technique for estimating the foreground and background motions in regions of reflections. Most of the existing motion estimators assume the presence of one motion per pel. Current existing multiple motion estimators assume constant motion over at least three frames. In addition they impose temporal consistency on the generated motions. As a result they cannot handle temporally active motion arising due to slight camera shake or acceleration.

We presented a novel approach for multiple motion estimation by treating the problem as a joint layer-motion estimation. This is based on Sarel et al. work [74] which showed that the motion of one layer can be used to temporally align this layer over the other moving layer. The static layer is then extracted using the Weiss et al. approach [94]. Hence we modeled motions of the underlying background/foreground layers as the ones generating the best reconstruction of those layers. We overcame the computational burden of this approach by generating motion candidates from the KLT tracks of the examined reflection. We generated a solution within a Bayesian framework. Unlike current techniques we imposed temporal consistency on the generated layers not on the generated motions.

Ground-truth comparisons showed that our approach handles temporally active motions better than current techniques. In addition it is computationally more efficient than existing techniques. Our technique however generates false estimates in regions not containing reflections. We showed how to discard such regions from the final motion estimates using masks generated from our reflection detection technique.

8.4 Future Work

8.4.1 Corruption Removal

A main issue that remains in the proposed corruption removal techniques is how to combine both spatial and temporal reconstruction for optimal missing data reconstruction. Spatial reconstruction should be favored in regions undergoing complicated motion while temporal reconstruction should be favored in regions undergoing simple motion. This spatio-temporal fusion scheme is controlled by few (four) main system parameters some of which were configured automatically

and others were fixed throughout the examined data. Results however showed that incorrect settings of those parameters could lead to severe reconstruction errors. With a massive database of archived footage being available, a large variety of material can be processed with our techniques. This may require a better approach for spatio-temporal fusion. An approach to do so is by using non-binary fusion instead of binary fusion. Our technique assigns either a spatial or a temporal solution for each examined pel i.e. binary spatio-temporal mixing. Hence future work will explore the possibility of improving reconstruction quality by generating a final solution that is a linear mixture between the spatial and temporal solutions i.e. non-binary mixing.

Generating more accurate motion estimates will make our removal techniques more robust to complicated motions. Motion estimation is a classical image processing problem [37, 48]. Pathological motion however still remains a major problem that prevents motion information to be used ubiquitously for video editing. Pathological motion comes in many forms e.g. occlusion, uncovering, motion blur [19]. Kokaram's blotch remover JONDI [47] attempted to bring more robustness to motion errors by incorporating fields that explicitly describe occlusion and uncovering. Future work will aim to improve our removal techniques by incorporating similar occlusion fields in our system.

8.4.2 Reflection Detection in Image Sequences

Reflection detection in image sequences is a new research area with the first technique presented in this thesis. We showed that layer separability can be used as a strong cue for reflection detection. Future work will aim to improve detection by improving layer separation quality. However as plenty of work already exists on layer separation from multiple images, future work will focus on layer separation from a single still image. More attention will be given to improving our still image layer separation technique.

So far our detection technique is trained over artificially created data. However as reflections come in a large variety of shapes and colors, future work will aim to generate better detection by training our technique over a large database of real data. This however will require painstaking manual ground-truth reflection detection. A further line of research is to improve the computational complexity of our reflection detector. This can be done by discarding the weak cues from the detection process.

Since our reflection detector is used as a preprocessing tool to detect regions of failure in current video processing techniques, lots of research can be carried out on improving the robustness of current video processing techniques to regions of reflections. Future work will focus on improving object recognition and motion estimation in regions of reflections.

8.4.3 Multiple Motion Estimation for Regions of Reflections

We showed that multiple motion estimation can be formulated as the joint problem of layer-motion estimation. Future work will aim to generate more accurate motion estimates by improv-

ing the quality of layer separation. Recall that we generated motion candidates from KLT tracks. Future work will investigate the possibility of generating motion candidates using an exhaustive search. This is expected to improve motion estimates however will increase computational complexity and may require modification of the energy terms. A joint reflection detection/motion estimation system can then be developed given good layer separation is generated.

One promising application to our multiple motion estimator is denoising regions of reflections. We will first extract the underlying layers and their motions using our motion estimator. Each layer will then be denoised separately. The noise-reduced sequence will then be generated by combining the denoised layers back together using the estimated motion vectors. Another application to our multiple motion estimator is enhancing the contrast of the layer of interest in medical images i.e. X-ray, MRI scans [7]. Such images often have multiple semi-transparent layers over each other. Hence the layer of interest can be extracted from the observed image mixture by temporally registering this layer using our motion estimator followed by temporal median filtering.

9

Appendix A: MAP Estimate of Bayesian Matting

The posterior for Bayesian Matting is as follows

$$P(\alpha, F, B|M) \propto \exp - \left(\frac{||M - \alpha F - (1-\alpha)B||^2}{2\sigma_e^2} + (F - \bar{\mathbf{F}})^T \mathbf{R}_F (F - \bar{\mathbf{F}}) + (B - \bar{\mathbf{B}})^T \mathbf{R}_B (B - \bar{\mathbf{B}}) \right) \tag{9.1}$$

Recall $[\bar{\mathbf{F}}, \mathbf{R}_F]$ and $[\bar{\mathbf{B}}, \mathbf{R}_B]$ are the mean and covariance of the examined foreground and background color distributions respectively. σ_e^2 is the variance of the error in the observation model and $[\alpha, F, B]$ are the unknown opacity, foreground and background colors respectively for the examined pel. The MAP estimate of $[\alpha, F, B]$ is calculated over three main stages: 1) B given $[F, \alpha]$ is estimated by differentiating equation 9.1 w.r.t B and equating to 0 2) F given $[B, \alpha]$ is estimated by differentiating equation 9.1 w.r.t F and equating to 0 and finally 3) α given $[F, B]$ is estimated by differentiating equation 9.1 w.r.t α and equating to 0. We discuss each stage in more details here. Note that for every examined site the parameters $[\sigma_e^2, \bar{\mathbf{F}}, \mathbf{R}_F, \bar{\mathbf{B}}, \mathbf{R}_B]$ are already known and hence are constants. All calculations are performed in the RGB color space

9.0.3.1 Estimating F given $[B, \alpha]$

$\frac{\partial}{\partial F} P(\alpha, F, B|M)$ can be simplified into three separate derivative operations as follows

$$\frac{\partial}{\partial F} P(\alpha, F, B|M) \propto \frac{\partial}{\partial F} \frac{||M - \alpha F - (1-\alpha)B||^2}{2\sigma_e^2} + \frac{\partial}{\partial F} (F - \bar{\mathbf{F}})^T \mathbf{R}_F (F - \bar{\mathbf{F}}) + \frac{\partial}{\partial B} (B - \bar{\mathbf{B}})^T \mathbf{R}_B (B - \bar{\mathbf{B}}) \tag{9.2}$$

Here the last derivate is 0 as B is constant w.r.t to F.

For one color channel, $\frac{\partial}{\partial F}\frac{||M-\alpha F-(1-\alpha)B||^2}{2\sigma_e^2}$ becomes

$$\frac{\partial}{\partial F}\frac{||M-\alpha F-(1-\alpha)B||^2}{2\sigma_e^2} = (\alpha^2 F + \alpha(1-\alpha)B - \alpha M)/\sigma_e^2 \tag{9.3}$$

To calculate $\frac{\partial}{\partial F}(F-\bar{\mathbf{F}})^T\mathbf{R}_F(F-\bar{\mathbf{F}})$ we do the following simplifications. Denote $\mathbf{R}_F$ by

$$\mathbf{R}_F = \begin{bmatrix} a_{rr} & a_{rg} & a_{rb} \\ a_{gr} & a_{gg} & a_{gb} \\ a_{br} & a_{bg} & a_{bb} \end{bmatrix} \tag{9.4}$$

where $a_{(.)}$ are the coefficients of the covariance matrix $\mathbf{R}_F$ and (r, g, b) denote the red, green and blue channels respectively. Hence we have

$$\begin{aligned}
&\frac{\partial}{\partial F}(F-\bar{\mathbf{F}})^T\mathbf{R}_F(F-\bar{\mathbf{F}}) \\
&= \frac{\partial}{\partial F}\begin{bmatrix} F_r-\bar{F_r} & F_g-\bar{F_g} & F_b-\bar{F_b} \end{bmatrix}\begin{bmatrix} a_{rr} & a_{rg} & a_{rb} \\ a_{gr} & a_{gg} & a_{gb} \\ a_{br} & a_{bg} & a_{bb} \end{bmatrix}\begin{bmatrix} F_R-\bar{F_R} \\ F_G-\bar{F_G} \\ F_B-\bar{F_B} \end{bmatrix} \\
&= \begin{bmatrix} F_r-\bar{F_r} & F_g-\bar{F_g} & F_b-\bar{F_b} \end{bmatrix}\begin{bmatrix} (F_r-\bar{F_r})a_{rr}+(F_g-\bar{F_g})a_{rg}+(F_b-\bar{F_b})a_{rb} \\ (F_r-\bar{F_r})a_{gr}+(F_g-\bar{F_g})a_{gg}+(F_b-\bar{F_b})a_{gb} \\ (F_r-\bar{F_r})a_{br}+(F_g-\bar{F_g})a_{bg}+(F_b-\bar{F_b})a_{bb} \end{bmatrix}
\end{aligned} \tag{9.5}$$

Multiplying the two vectors of equation 9.5 together, differentiating w.r.t the red, green and blue channels of F and grouping similar terms together, $\frac{\partial}{\partial F}(F-\bar{\mathbf{F}})^T\mathbf{R}_F(F-\bar{\mathbf{F}})$ becomes

$$\frac{\partial}{\partial F_r}(F-\bar{\mathbf{F}})^T\mathbf{R}_F(F-\bar{\mathbf{F}}) = (F_r-\bar{F_r})a_{11}+(F_g-\bar{F_g})a_{12}+(F_b-\bar{F_b})a_{13} \tag{9.6}$$

$$\frac{\partial}{\partial F_g}(F-\bar{\mathbf{F}})^T\mathbf{R}_F(F-\bar{\mathbf{F}}) = (F_r-\bar{F_r})a_{21}+(F_g-\bar{F_g})a_{22}+(F_b-\bar{F_b})a_{23} \tag{9.7}$$

$$\frac{\partial}{\partial F_r}(F-\bar{\mathbf{F}})^T\mathbf{R}_F(F-\bar{\mathbf{F}}) = (F_r-\bar{F_r})a_{31}+(F_g-\bar{F_g})a_{32}+(F_b-\bar{F_b})a_{33} \tag{9.8}$$

Adding every equation from equations 9.6-9.8 with its corresponding color component of equation. 9.3 and equating the final expressions to zero to maximize $P(F|\alpha, B, M)$ w.r.t F, we get

$$\begin{bmatrix} \sigma_e^2\mathbf{R}_F^{-1}+I\alpha^2 & I\alpha(1-\alpha) \end{bmatrix}\begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \sigma_e^2\mathbf{R}_F^{-1}\bar{\mathbf{F}}+\alpha M \end{bmatrix} \tag{9.9}$$

Here I is the 3×3 Identity matrix. That is a system of three equations and three unknowns being the RGB channels of F. This system solves for the optimal value of F given $[B, \alpha]$.

9.0.3.2 Estimating B given $[F, \alpha]$

$\frac{\partial}{\partial B}P(\alpha, F, B|M)$ is obtained in a similar way to $\frac{\partial}{\partial F}P(\alpha, F, B|M)$. By analogy, $\frac{\partial}{\partial B}P(\alpha, F, B|M)$ is obtained by substituting every F and α in equation 9.9 with B and $1-\alpha$ respectively. This

generates

$$\begin{bmatrix} I\alpha(1-\alpha) & \sigma_e^2\mathbf{R}_B^{-1} + I(1-\alpha)^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \sigma_e^2\mathbf{R}_B^{-1}\bar{\mathbf{B}} + (1-\alpha)M \end{bmatrix} \tag{9.10}$$

where I is the 3×3 Identity matrix. This system solves for the optimal value of B given $[F, \alpha]$.

9.0.3.3 Estimating α given $[F, B]$

Differentiating $P(\alpha, F, B|M)$ w.r.t to α amounts to differentiating just the first term of equation 9.1 as the last two terms derivatives w.r.t α are 0. To calculate $\frac{\partial}{\partial\alpha}\frac{||M-\alpha F-(1-\alpha)B||^2}{2\sigma_e^2}$ we first expand $\frac{||M-\alpha F-(1-\alpha)B||^2}{2\sigma_e^2}$ as follows

$$\begin{aligned} \frac{||M-\alpha F-(1-\alpha)B||^2}{2\sigma_e^2} = ((M_r - \alpha F_r - (1-\alpha)B_r)^2 + \\ (M_g - \alpha F_g - (1-\alpha)B_g)^2 + (M_b - \alpha F_b - (1-\alpha)B_b)^2)/2\sigma_e^2 \end{aligned} \tag{9.11}$$

where $[r, g, b]$ are the red, green and blue components respectively. To maximize $P(\alpha, F, B|M)$ w.r.t to α we differentiate the right hand side of equation 9.11 w.r.t to α and equate the resulting expression to 0. This leads to

$$\begin{aligned} \alpha((F_r - B_r)^2 + (F_g - B_g)^2 + (F_b - B_b)^2) = (F_r - B_r)(M_r - B_r) \\ + (F_g - B_g)(M_g - B_g) + (F_g - B_g)(M_g - B_g) \end{aligned} \tag{9.12}$$

Grouping α in one side we get

$$\alpha = \frac{(M-B)^T(F-B)}{||F-B||^2} \tag{9.13}$$

Equation. 9.13 solves for the optimal value of α given $[F, B]$.

The final set of equation in Bayesian Matting are Equations 9.9,9.10,9.13 which are grouped together for clarity as follows

$$\begin{bmatrix} \sigma_e^2\mathbf{R}_F^{-1} + I\alpha^2 & I\alpha(1-\alpha) \\ I\alpha(1-\alpha) & \sigma_e^2\mathbf{R}_B^{-1} + I(1-\alpha)^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \sigma_e^2\mathbf{R}_F^{-1}\bar{\mathbf{F}} + \alpha M \\ \sigma_e^2\mathbf{R}_B^{-1}\bar{\mathbf{B}} + (1-\alpha)M \end{bmatrix} \tag{9.14}$$

$$\alpha = \frac{(M-B)^T(F-B)}{||F-B||^2} \tag{9.15}$$

10

Appendix B: Artificial Reflection Databases

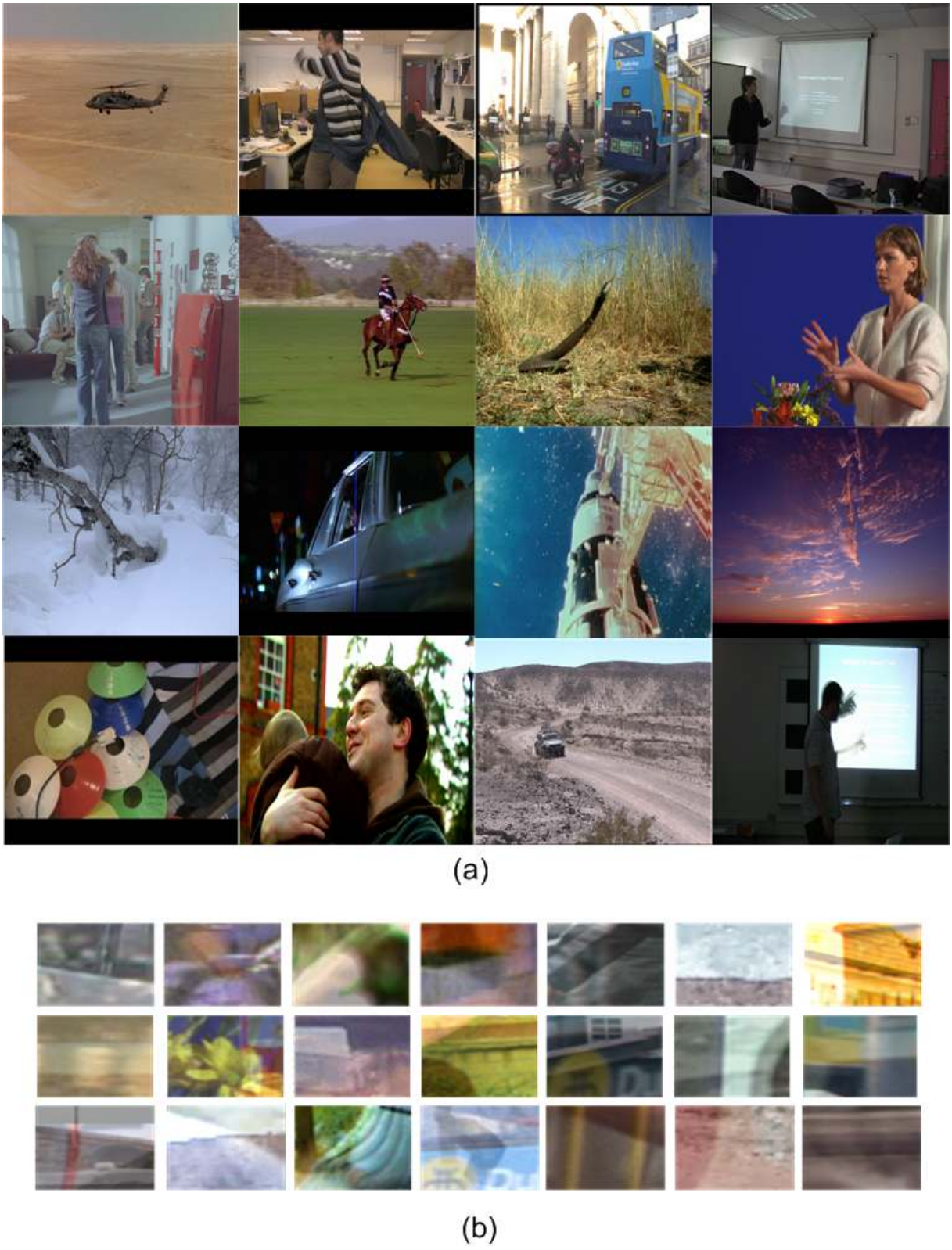


Figure 10.1: *(a) Frames from the database SynRef1. Such frames are used to create artificial reflections. (b) Examples of artificially created reflections from SynRef1.*

Figure 10.2: *Frames from the database SynRef2. Artificially created reflections are shown in blue*

11

Appendix C: Reflection Detection Results

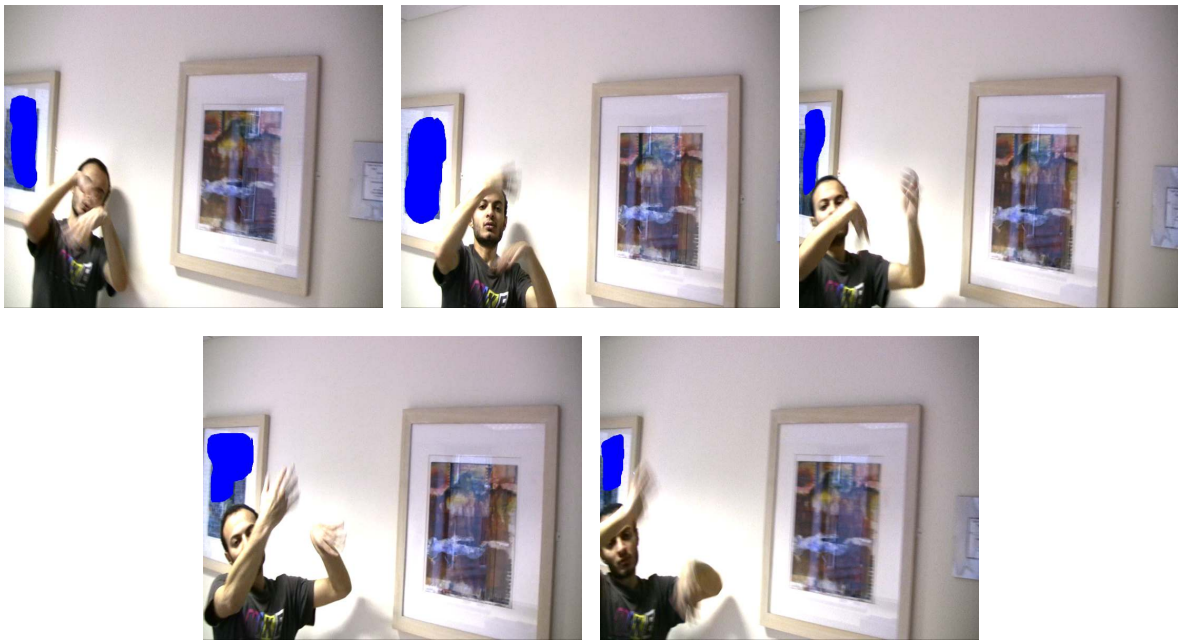


Figure 11.1: *Detection masks (shown in blue) generated manually for* ***SelimK2****. Those masks are used to detect the left picture frame in* ***SelimK2*** *(see figure 11.2, first row, shown in purple). SelimK2 contains 99 frames. Masks here are supplied at frames (in clockwise direction) 1, 30, 42, 75 and 85 respectively.*

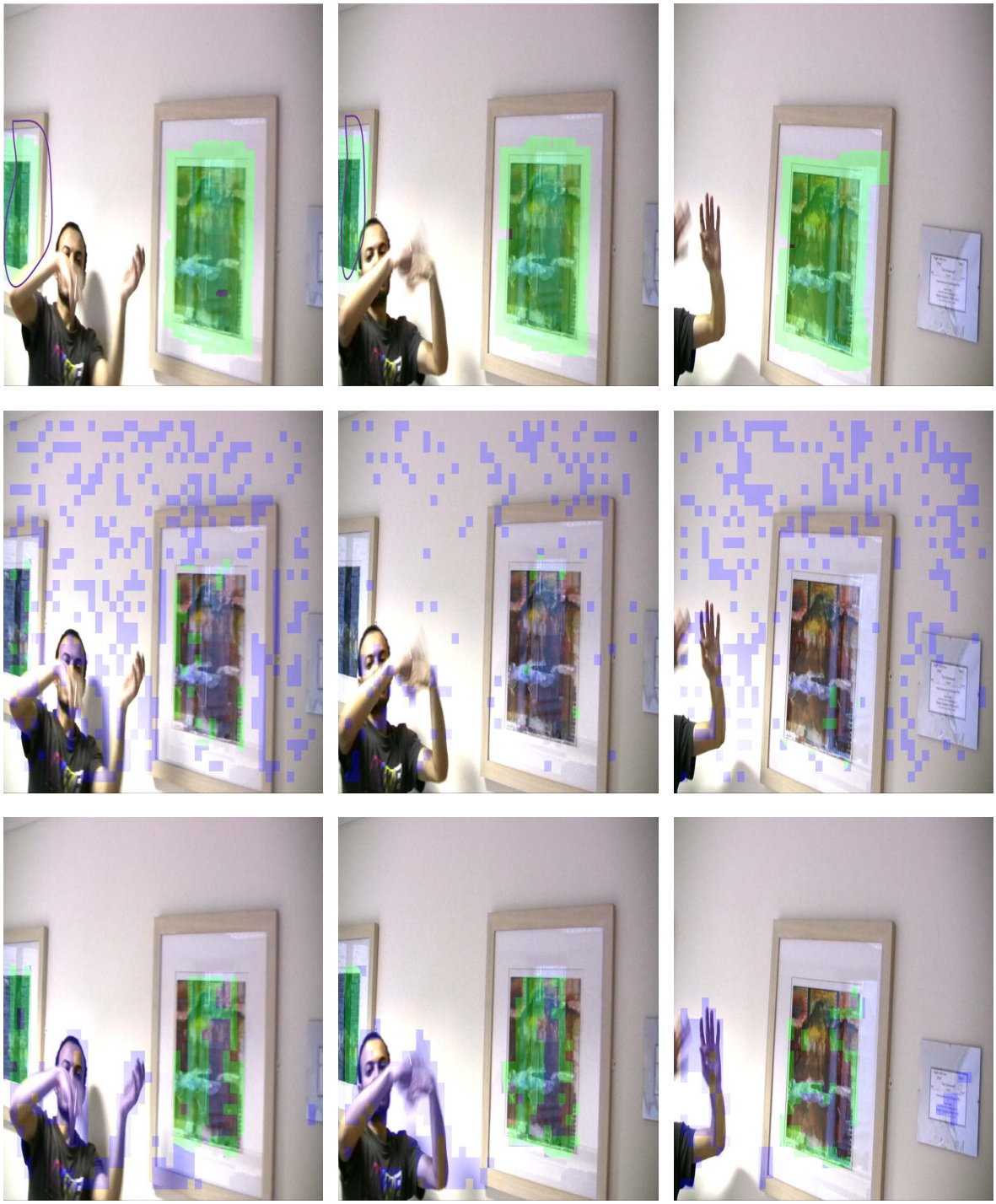


Figure 11.2: *Reflection detection on* ***SelimK2*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in blue. FEAPARD-F rejects pathological motion generated by the actor's hands. This motion however is detected by DFD. The picture frame on the left (first row, shown in purple) is detected using the manually supplied detection masks (see figure. 11.1).*

Figure 11.3: *Reflection detection on* ***BuildOnWind1*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. SHARPNESS misclassifies flat areas as reflection.*

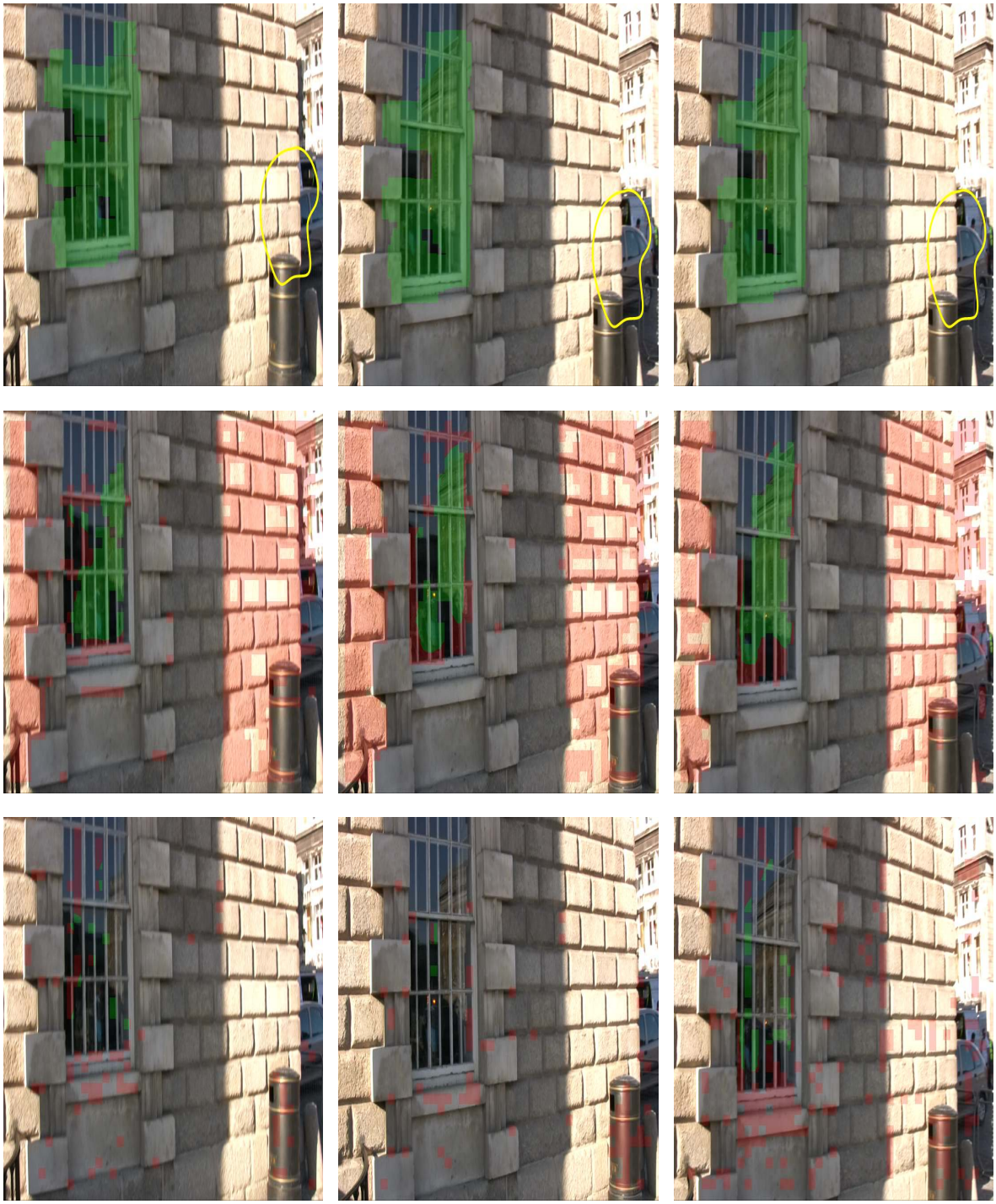


Figure 11.4: *Reflection detection on* ***BuildOnWind2*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. SHARPNESS misclassifies flat areas as reflections and DFD misclassifies regions of occlusion (first row, shown in yellow) as reflection. FEAPARD-F however rejects flat areas and regions of occlusion.*

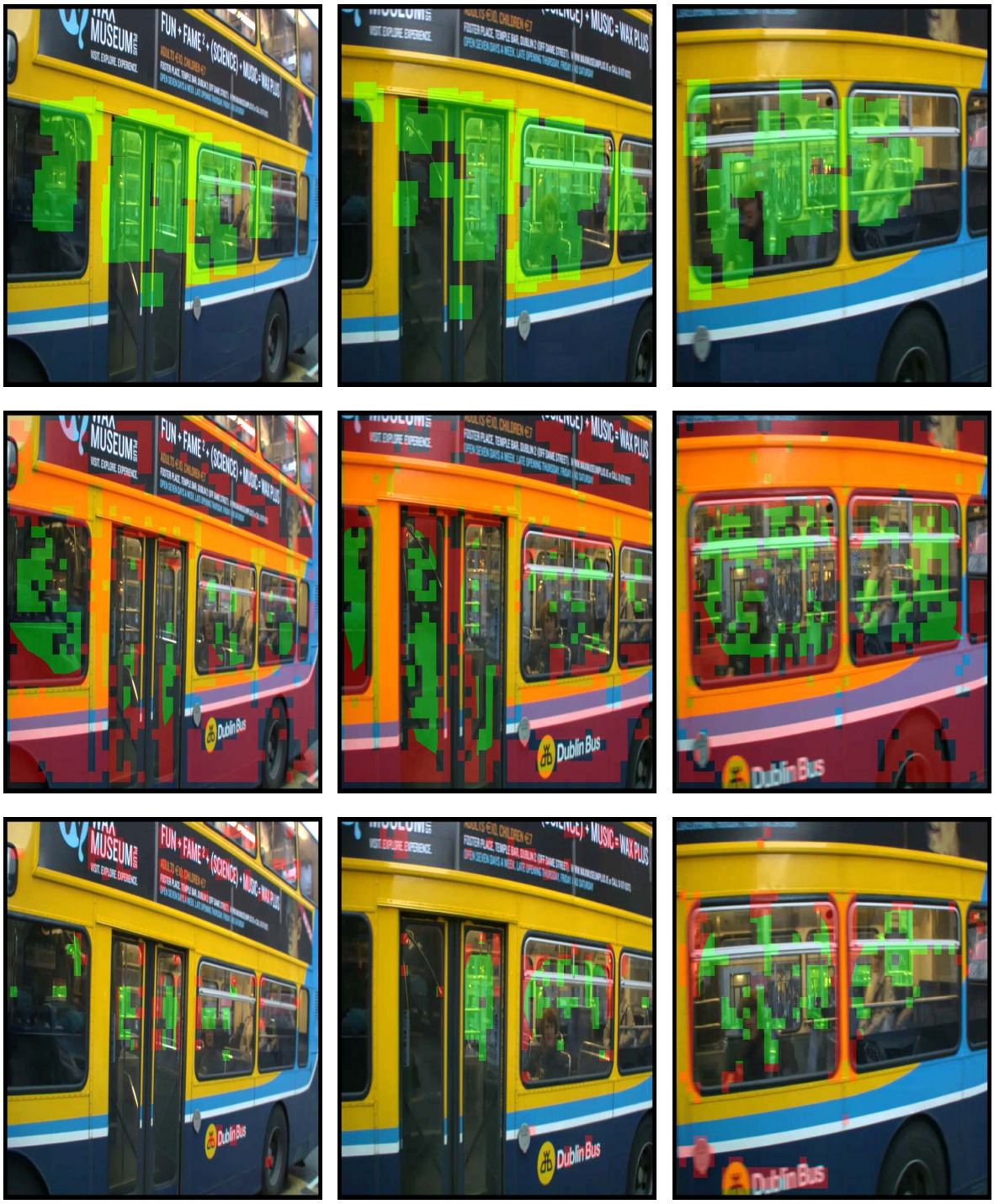


Figure 11.5: *Reflection detection on* ***Bus*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. SHARPNESS incorrectly flags flat areas as reflections and DFD generates some false alarms. FEAPARD-F however does not generate noticeable false detections.*

Figure 11.6: *Reflection detection on* ***Bulb*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in blue. SHARPNESS incorrectly flags flat areas as reflection.*

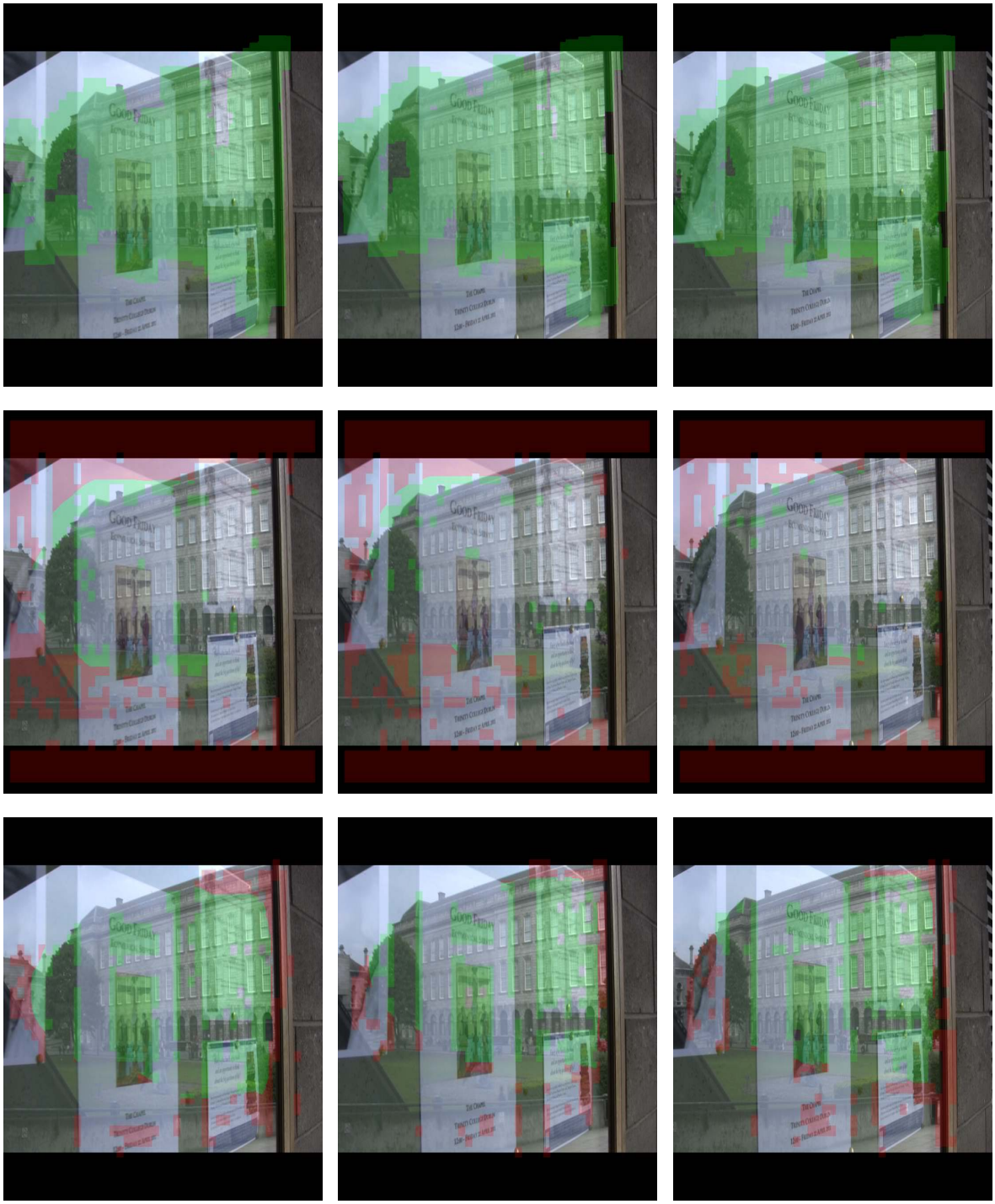


Figure 11.7: *Reflection detection on* ***BuildOnWind3*** *using, from top; FEAPARD-F, SHARPNESS and DFD respectively. Correct detections are shown in green while false detections are shown in red. SHARPNESS incorrectly flags flat areas as reflection.*

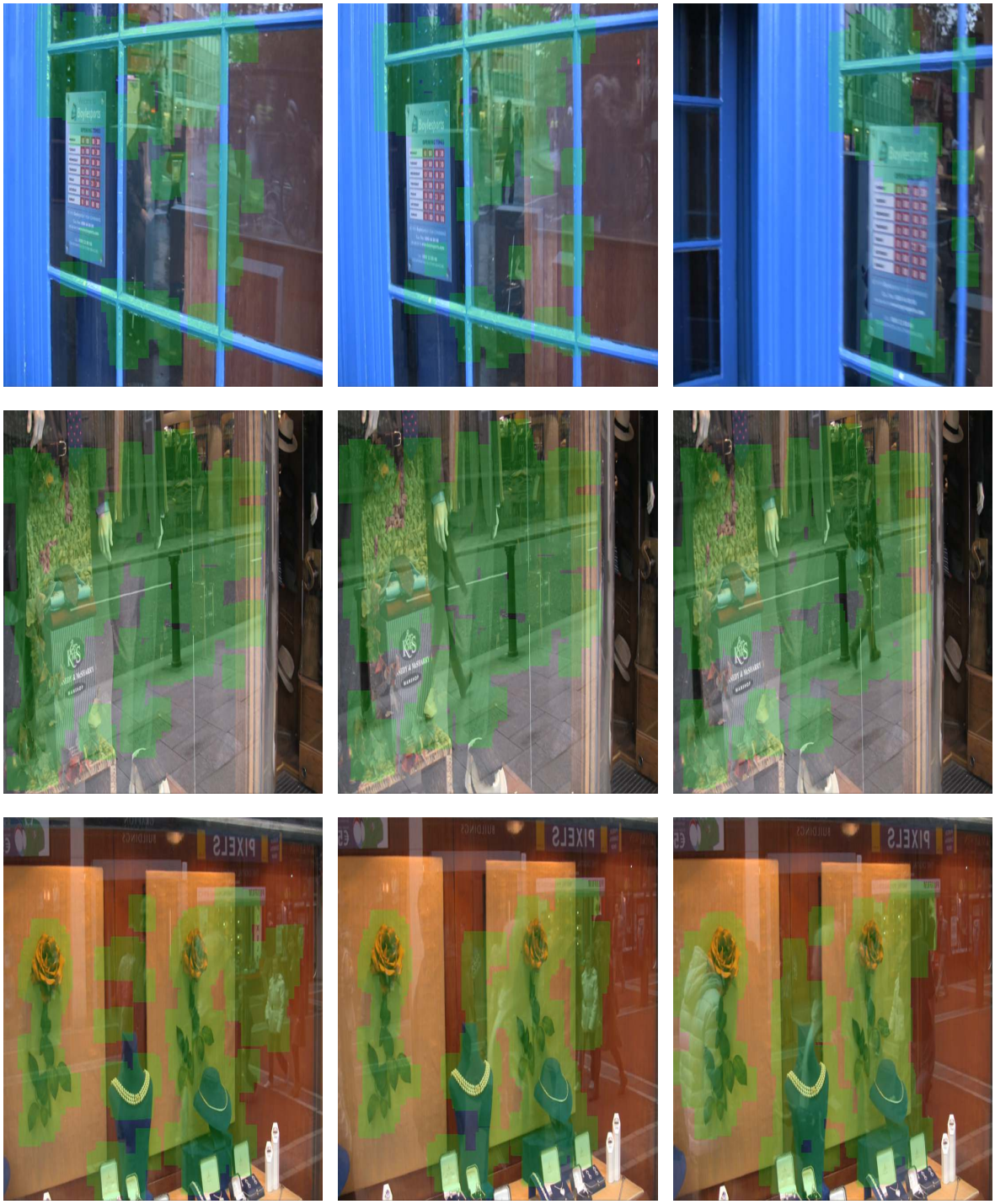


Figure 11.8: *Three frames of FEAPARD-F detection for, from top;* ***BluWind****,* ***GirlRef*** *and* ***RedPPL****. Correct detections are shown in green.*

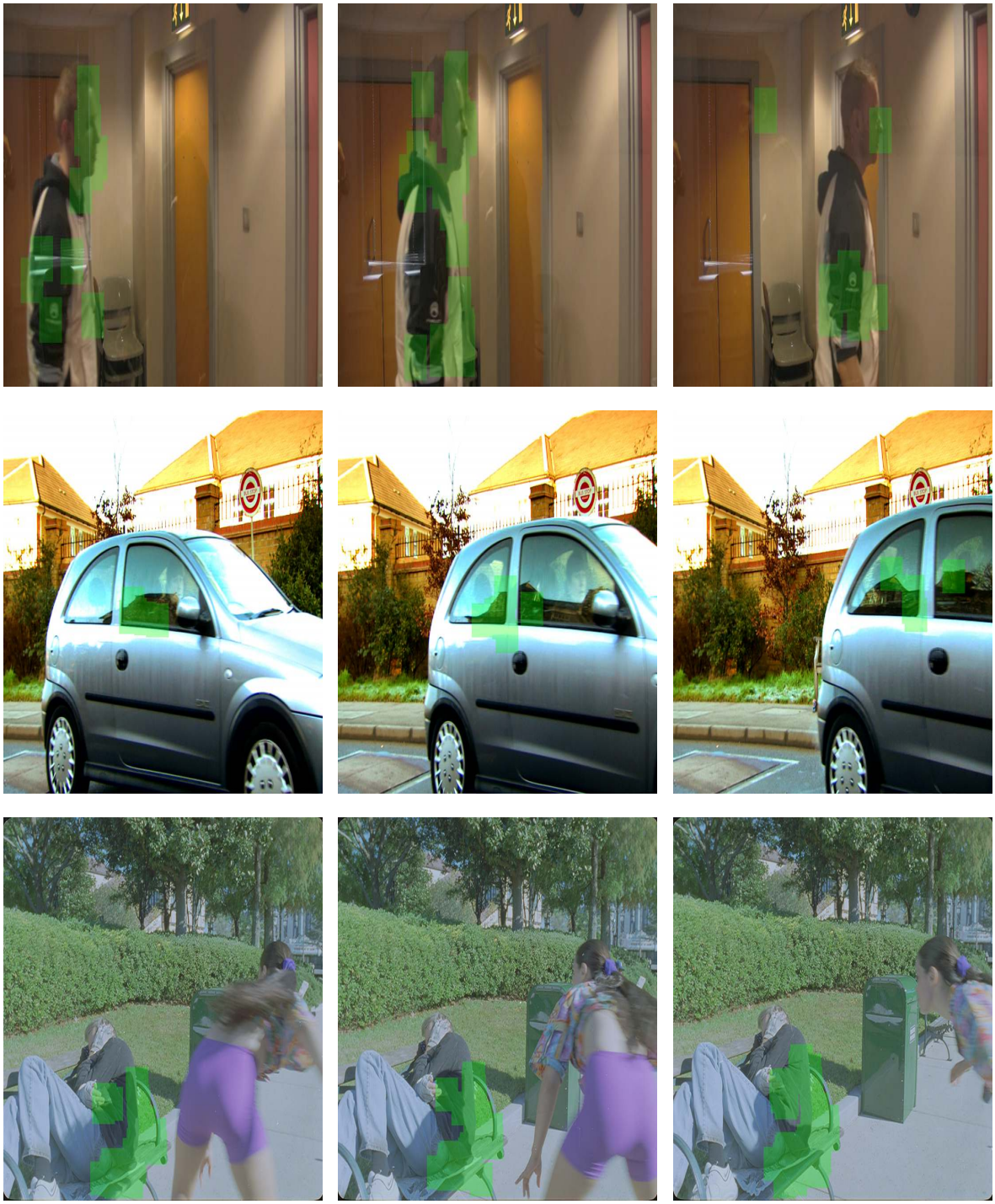


Figure 11.9: *Three frames of FEAPARD-F detection for, from top;* ***ManWalking****,* ***CarRef*** *and* ***GirlShadow****. Correct detections are shown in green.*

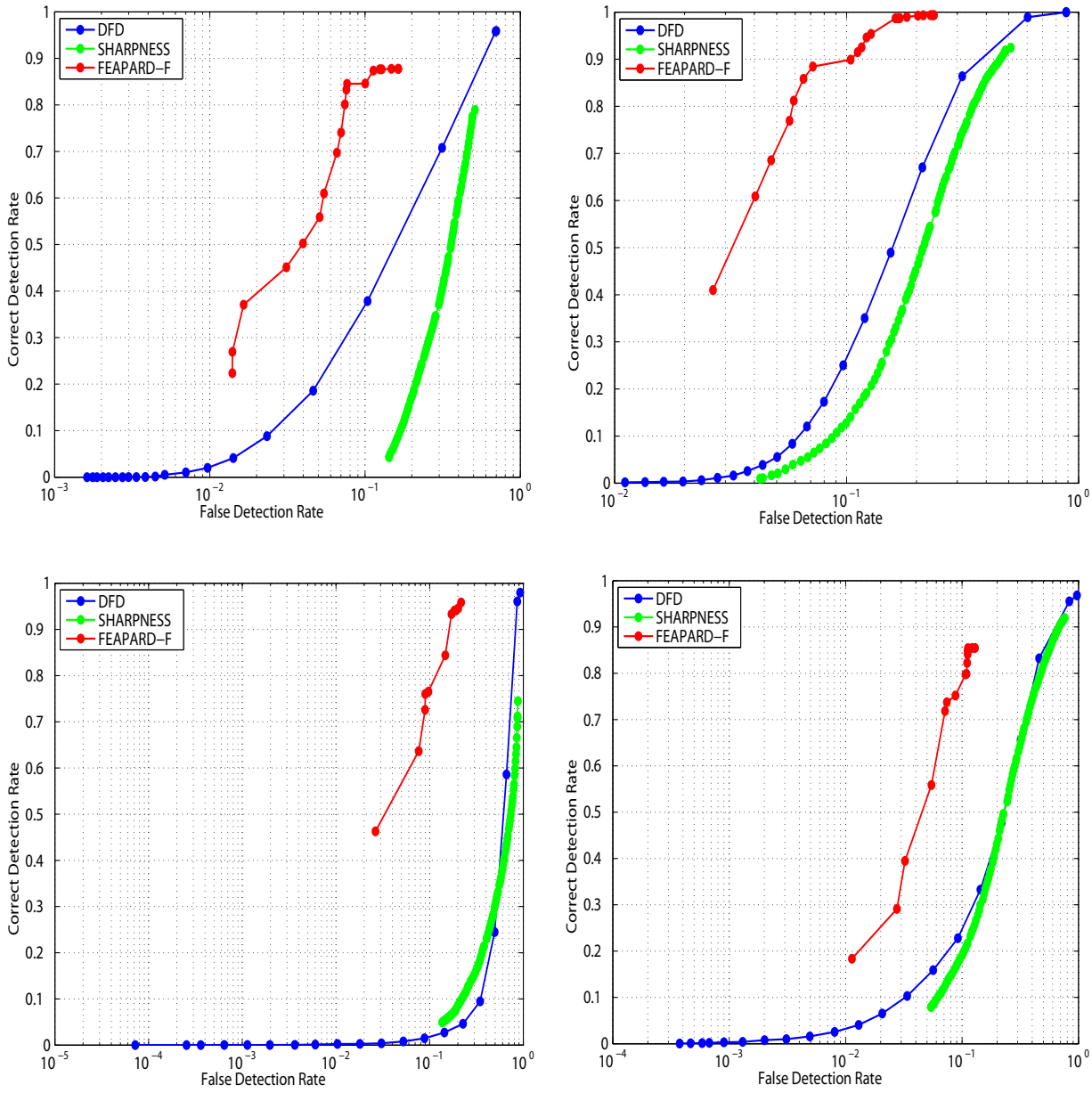


Figure 11.10: *ROC of FEAPARD-F, SHARPNESS and DFD for (in clockwise direction);* ***SelimH***, ***SelimC***, ***BuildOnWind2*** *and* ***BuildOnWind1*** *respectively. Our technique outperforms the other detectors with a massive increase in correct detection rate.*

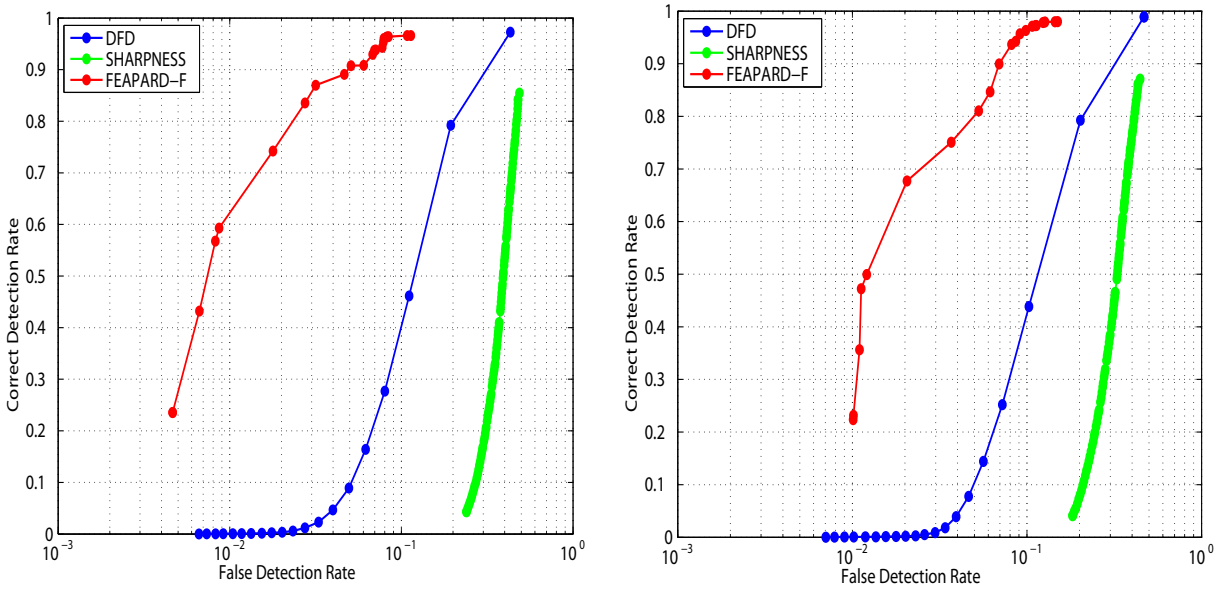


Figure 11.11: *ROC of FEAPARD-F, SHARPNESS and DFD for (from left);* ***SelimK1*** *and* ***SelimK2*** *respectively. FEAPARD-F outperforms DFD and SHARPNESS with a massive increase in correct detection rate.*

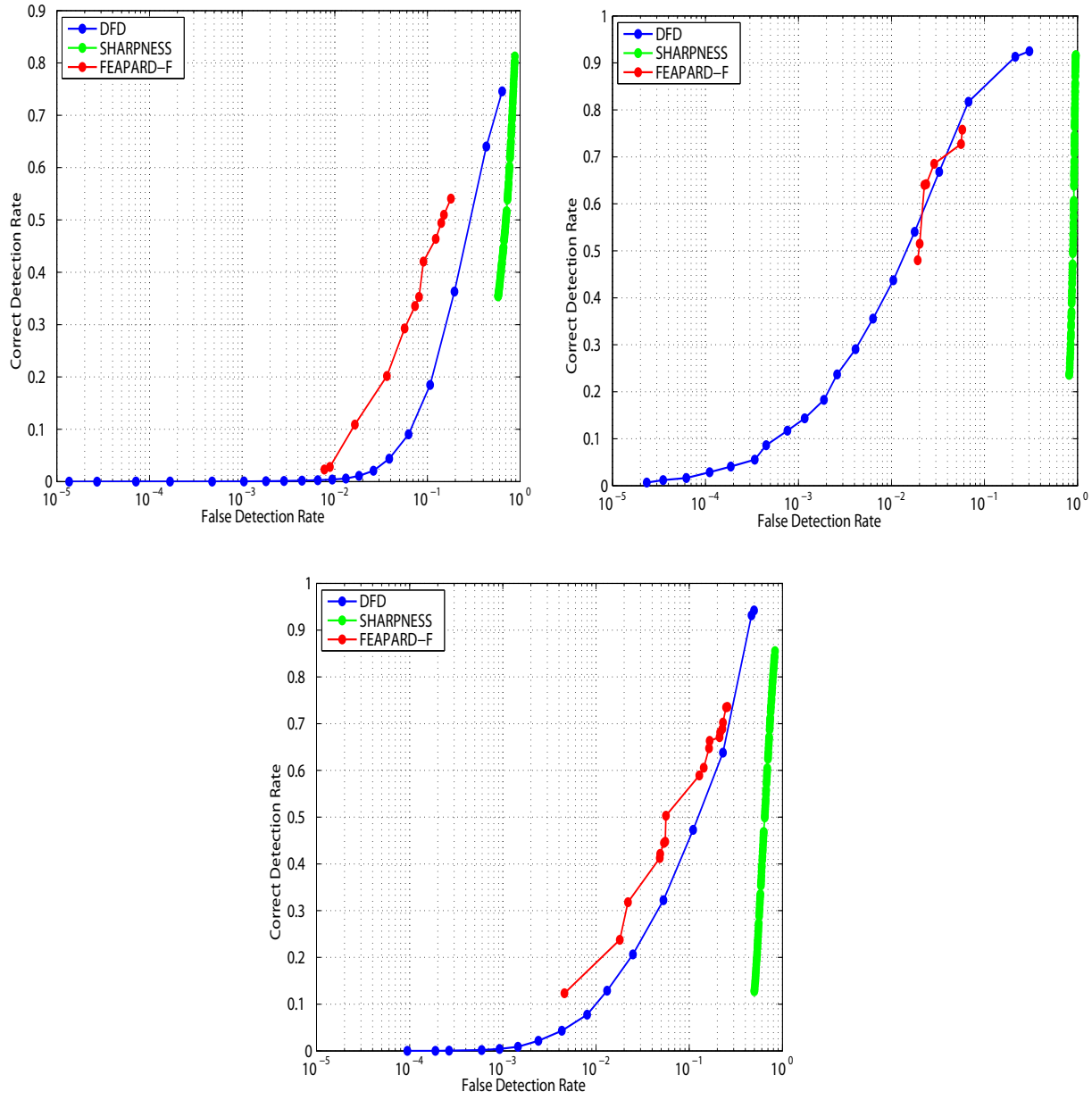


Figure 11.12: *ROC of FEAPARD-F, SHARPNESS and DFD for (in clockwise direction);* ***Bus****,* ***Bulb*** *and* ***BuildOnWind3*** *respectively. Our technique outperforms the other detectors.*

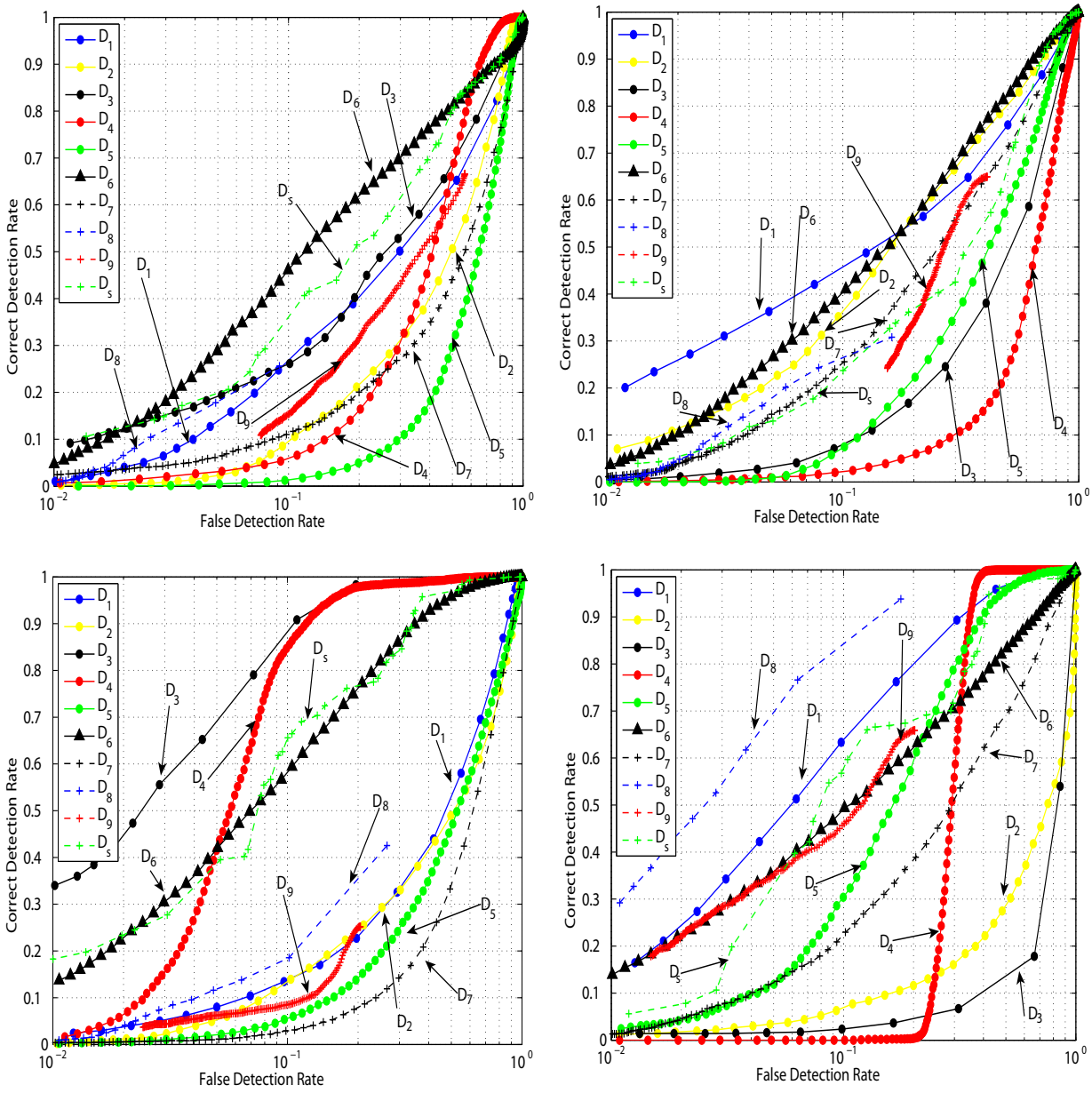


Figure 11.13: *ROC of $\mathcal{D}_{1-9}$ and $\mathcal{D}_s$ for (in clockwise direction);* ***SelimC***, ***SelimH***, ***BuildOnWind2*** *and* ***BuildOnWind1****. Either $\mathcal{D}_1$ or $\mathcal{D}_s$ outperform most of the remaining detectors in the examined sequences.*

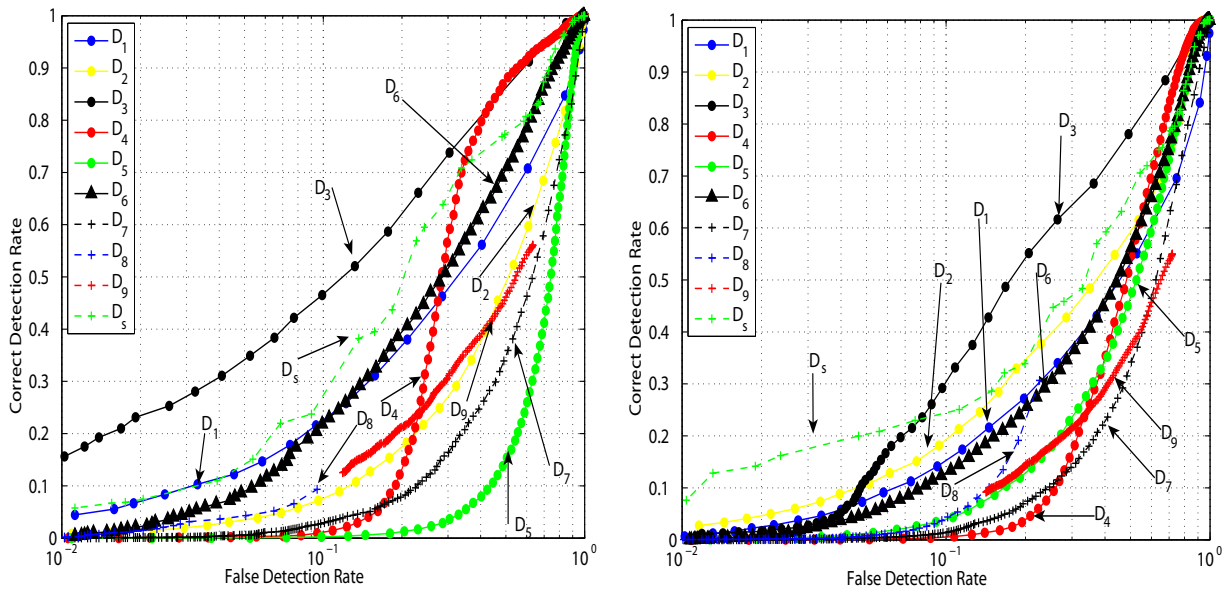


Figure 11.14: *ROC of $\mathcal{D}_{1-9}$ and $\mathcal{D}_s$ for (from left);* ***SelimK2*** *and* ***SelimK1****. $\mathcal{D}_s$ outperform most of the remaining detectors in both sequences.*

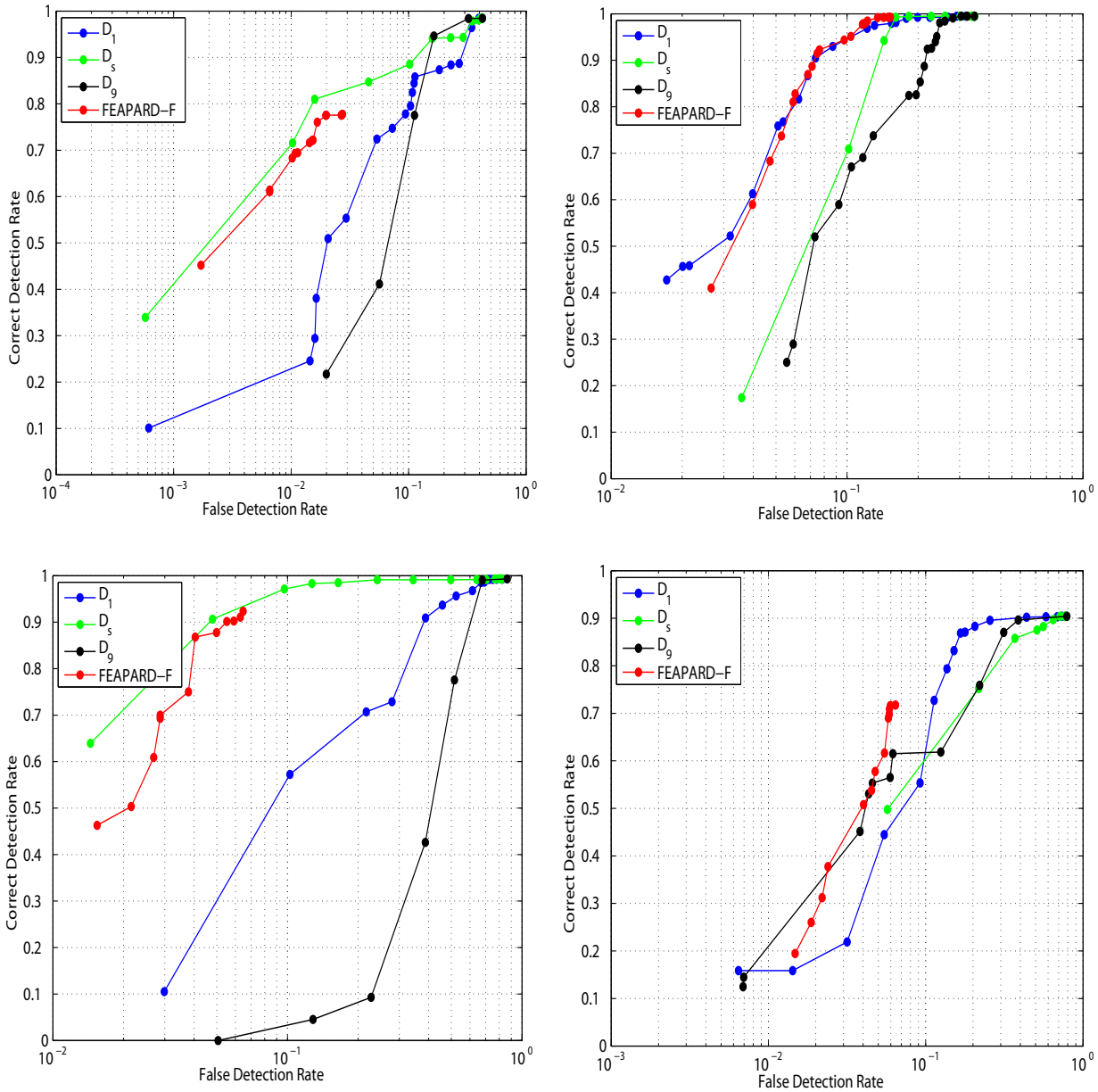


Figure 11.15: *ROC of $\mathcal{D}_1$, $\mathcal{D}_s$, $\mathcal{D}_9$, and FEAPARD-F as generated for (in clockwise direction);* ***SelimH***, ***SelimC***, ***BuildOnWind2*** *and* ***BuildOnWind1***. *Here the detection of $\mathcal{D}_1$ is obtained by removing $\mathcal{D}_s$ and $\mathcal{D}_9$ from the FEAPARD-F technique. The detections of $\mathcal{D}_s$ and $\mathcal{D}_9$ are obtained in a similar way. FEAPARD-F combines $\mathcal{D}_1$, $\mathcal{D}_s$ and $\mathcal{D}_9$ for optimal reflection detection.*

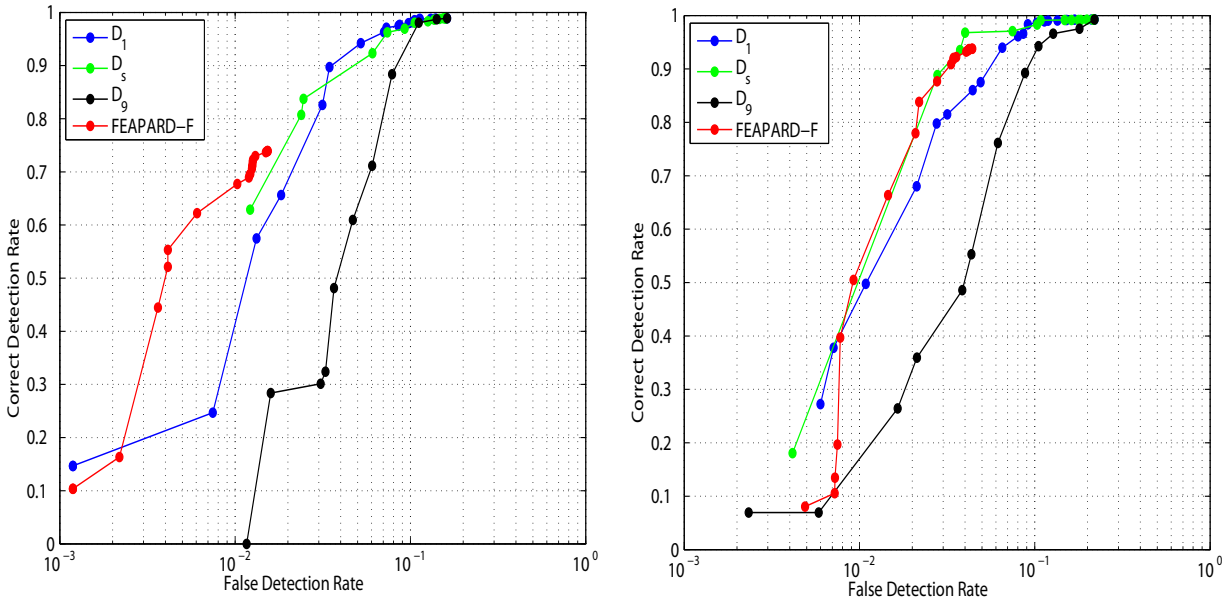


Figure 11.16: *ROC of $\mathcal{D}_1$, $\mathcal{D}_s$, $\mathcal{D}_9$, and FEAPARD-F as generated for (from left);* ***SelimK2*** *and* ***SelimK1*** *respectively. Here the detection of $\mathcal{D}_1$ is obtained by removing $\mathcal{D}_s$ and $\mathcal{D}_9$ from the FEAPARD-F technique. The detections of $\mathcal{D}_s$ and $\mathcal{D}_9$ are obtained in a similar way. FEAPARD-F combines $\mathcal{D}_1$, $\mathcal{D}_s$ and $\mathcal{D}_9$ for optimal reflection detection.*

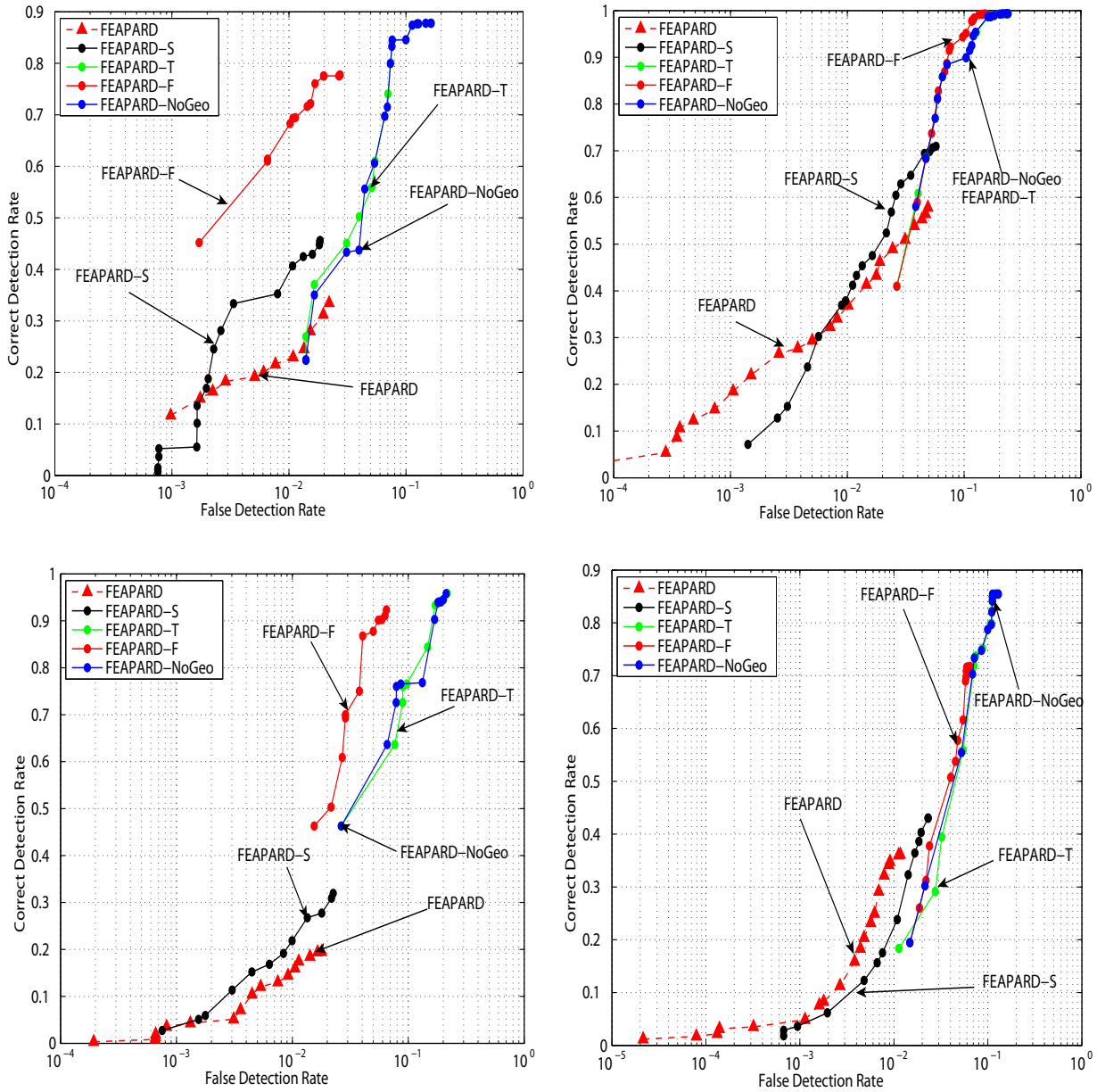


Figure 11.17: *ROC of different techniques generated for (in clockwise direction);* ***SelimH****,* ***SelimC****,* ***BuildOnWind2*** *and* ***BuildOnWind1****. FEAPARD is reflection detection without incorporating any spatial or temporal smoothness, while FEAPARD-S and FEAPARD-T is reflection detection but with incorporating only spatial and only temporal smoothness respectively. In FEAPARD-NoGeo we remove the geodesic distance from the spatial information of FEAPARD-F. The geodesic distance helps in rejecting false detections of FEAPARD. In addition, FEAPARD-F combines both spatial and temporal information for optimal detection.*

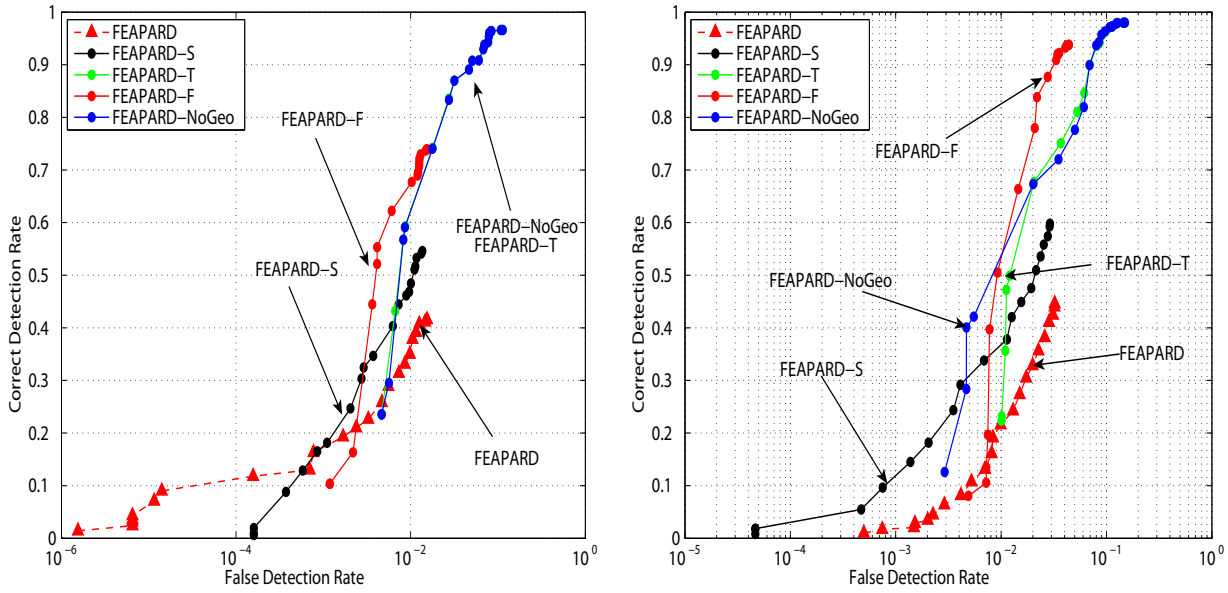


Figure 11.18: *ROC of different techniques generated for (from left);* ***SelimK2*** *and* ***SelimK1****. FEAPARD is reflection detection without incorporating any spatial or temporal smoothness, while FEAPARD-S and FEAPARD-T is reflection detection but with incorporating only spatial and only temporal smoothness respectively. In FEAPARD-NoGeo we remove the geodesic distance from the spatial information of FEAPARD-F. The geodesic distance helps in rejecting false detections of FEAPARD. In addition, FEAPARD-F combines both spatial and temporal information for optimal detection.*

12

Appendix D: Motion Estimation Results

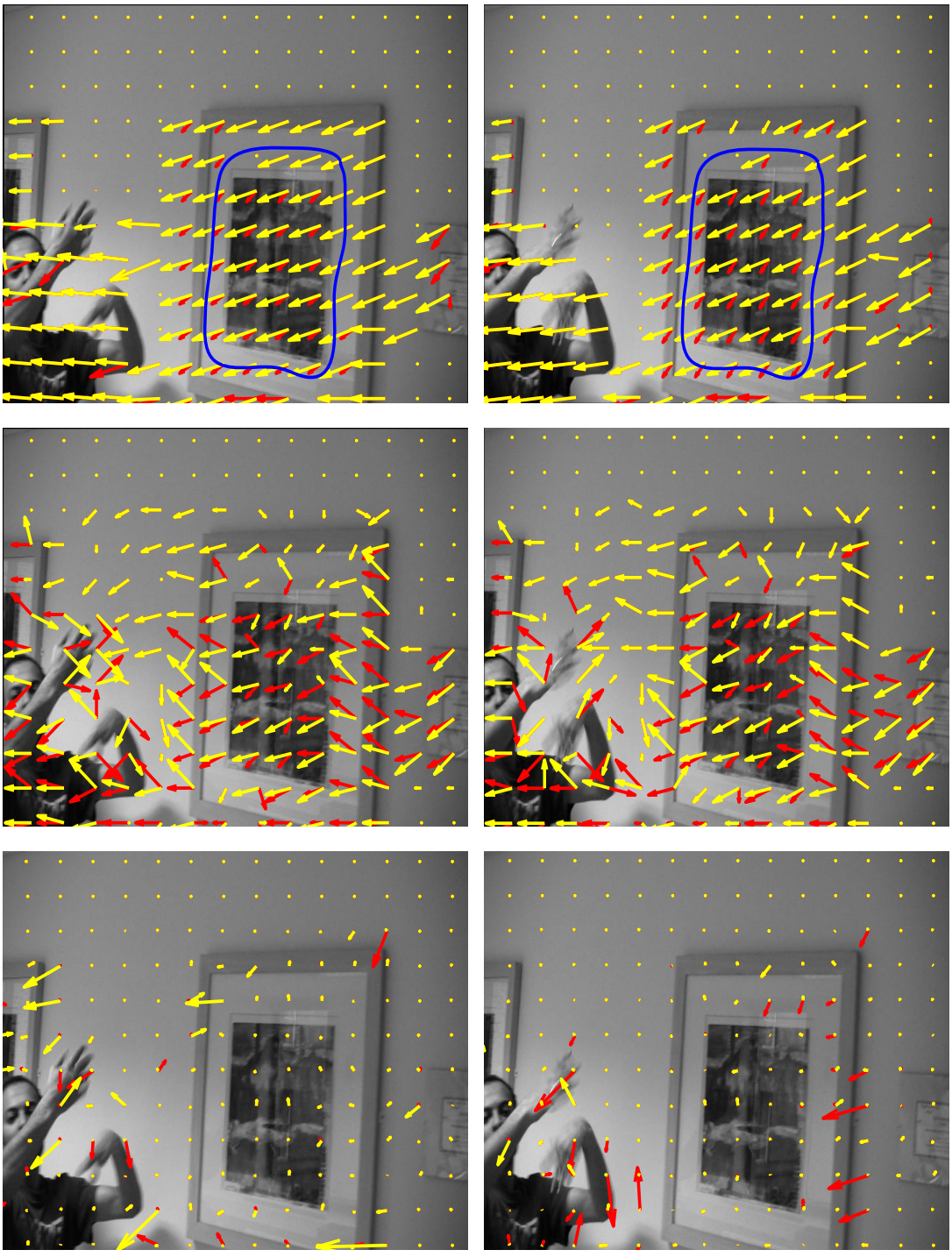


Figure 12.1: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 44 (left) and 45 (right) of* ***SelimK2****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 5 for illustration clarity. Regions of interest/reflections are shown in blue.*

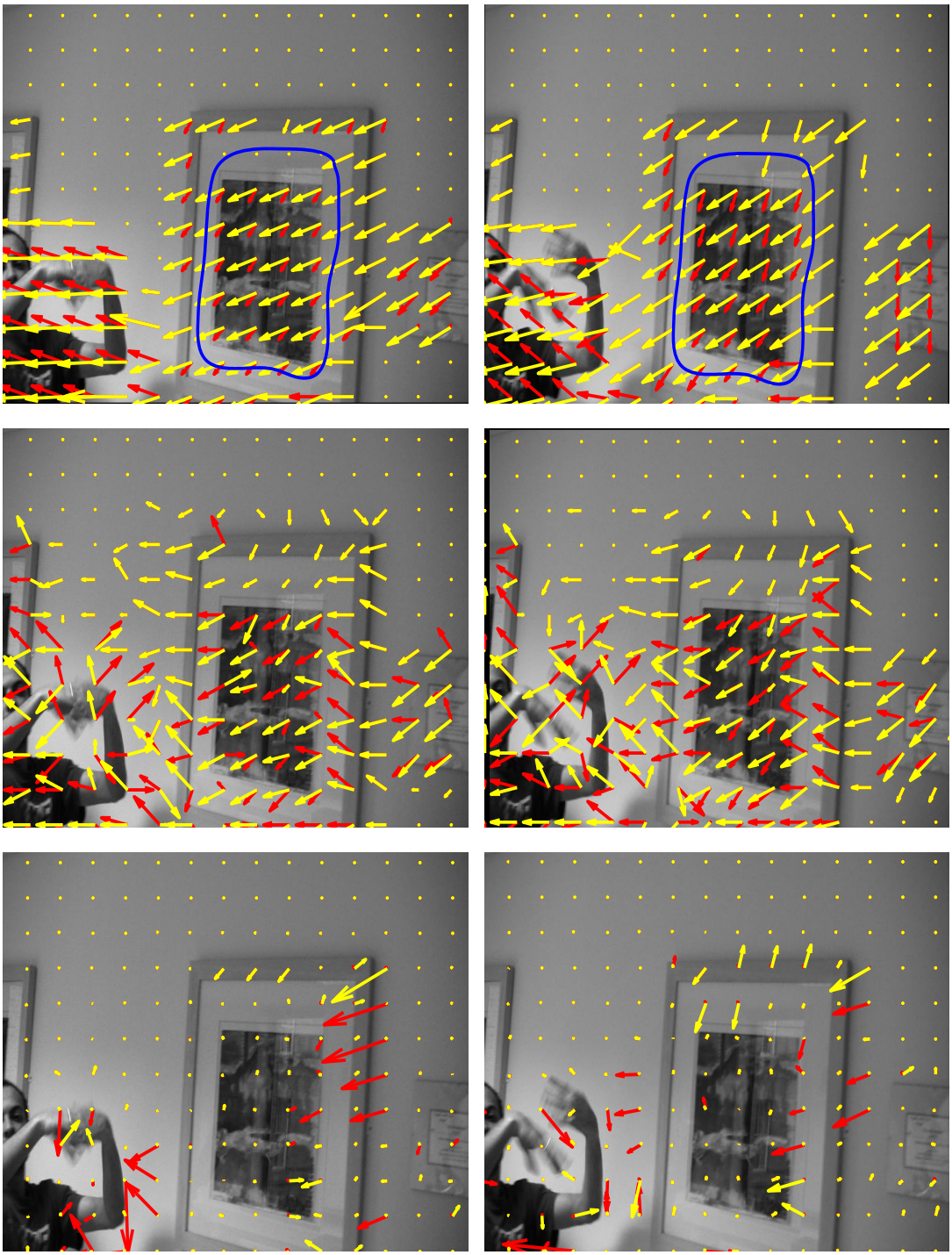


Figure 12.2: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 46 (left) and 47 (right) of* ***SelimK2****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 5 for illustration clarity. Regions of interest/reflections are shown in blue.*

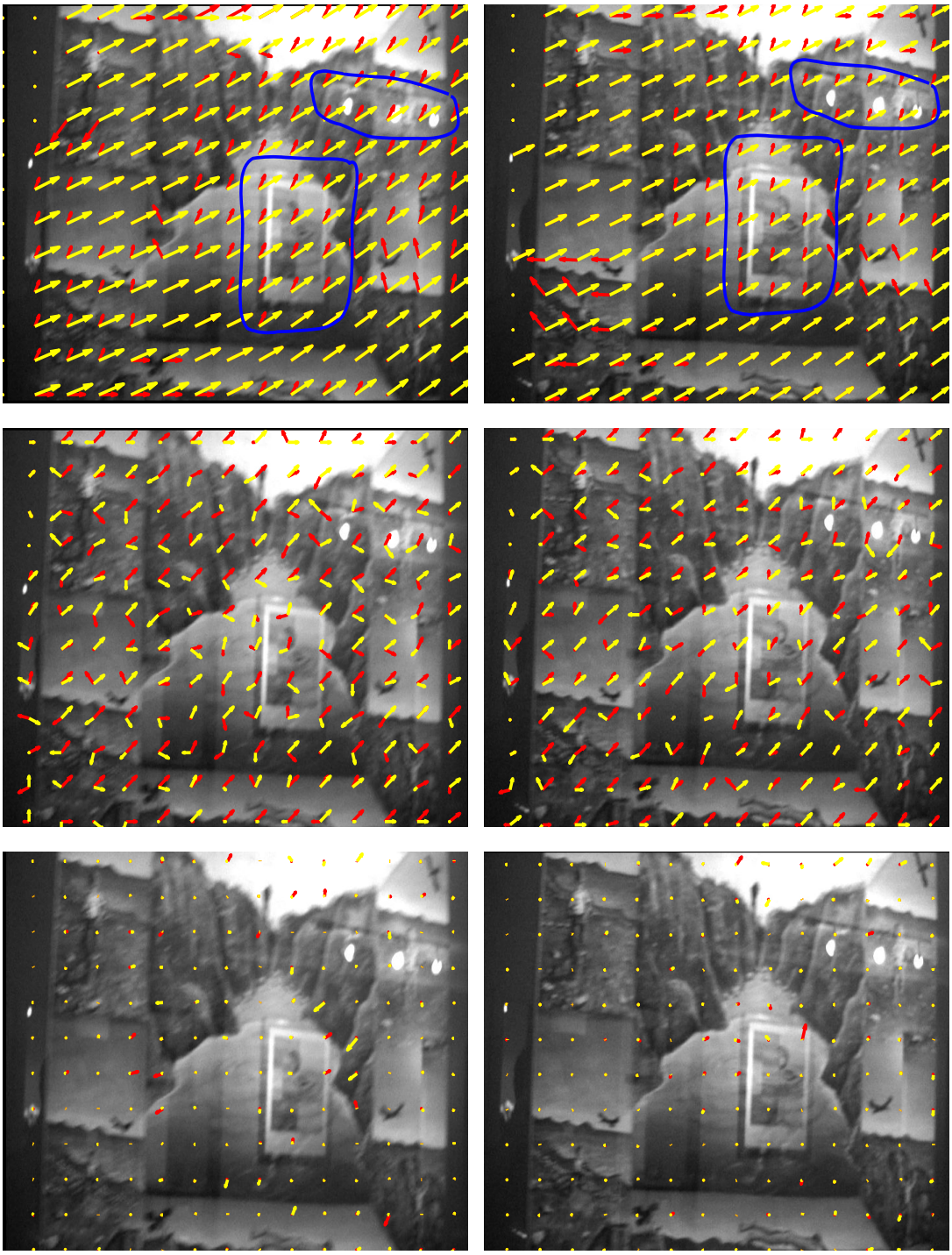


Figure 12.3: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 27 (left) and 28 (right) of* ***PicRef****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 2 for illustration clarity. Regions of interest/reflections are shown in blue.*

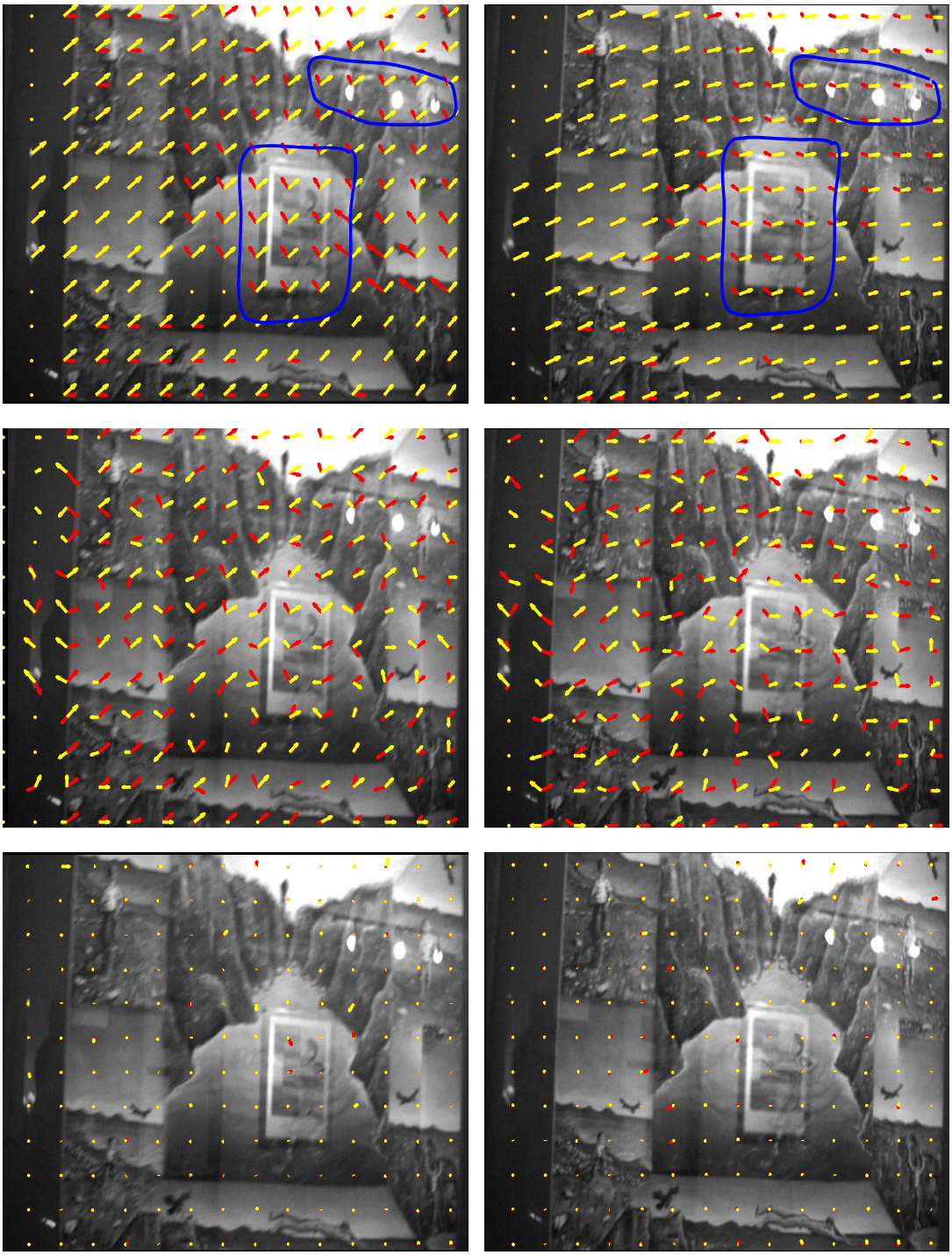


Figure 12.4: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 29 (left) and 30 (right) of* ***PicRef****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 2 for illustration clarity. Regions of interest/reflections are shown in blue.*

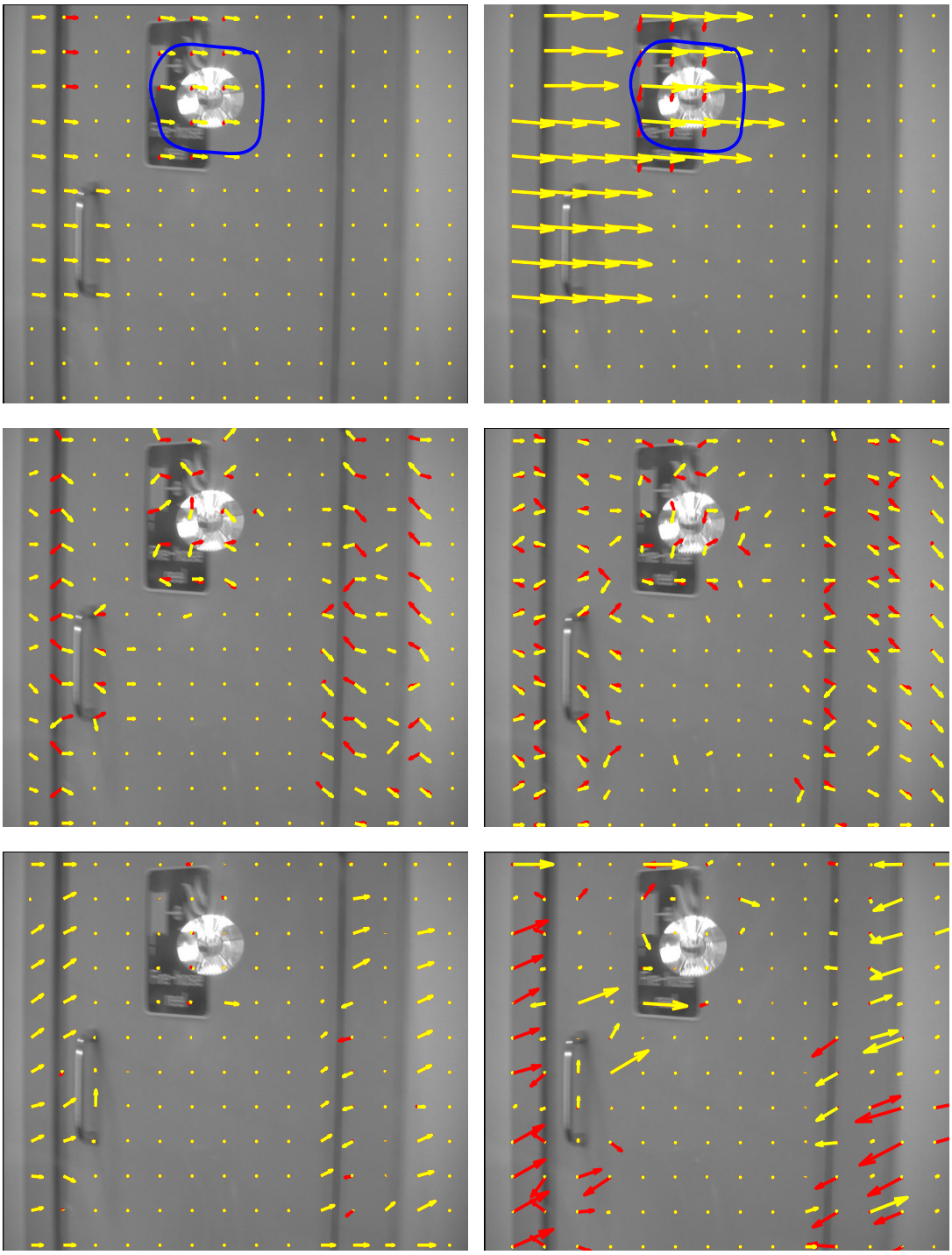


Figure 12.5: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 5 (left) and 6 (right) of* ***Bulb****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 2 for illustration clarity. Regions of interest/reflections are shown in blue.*

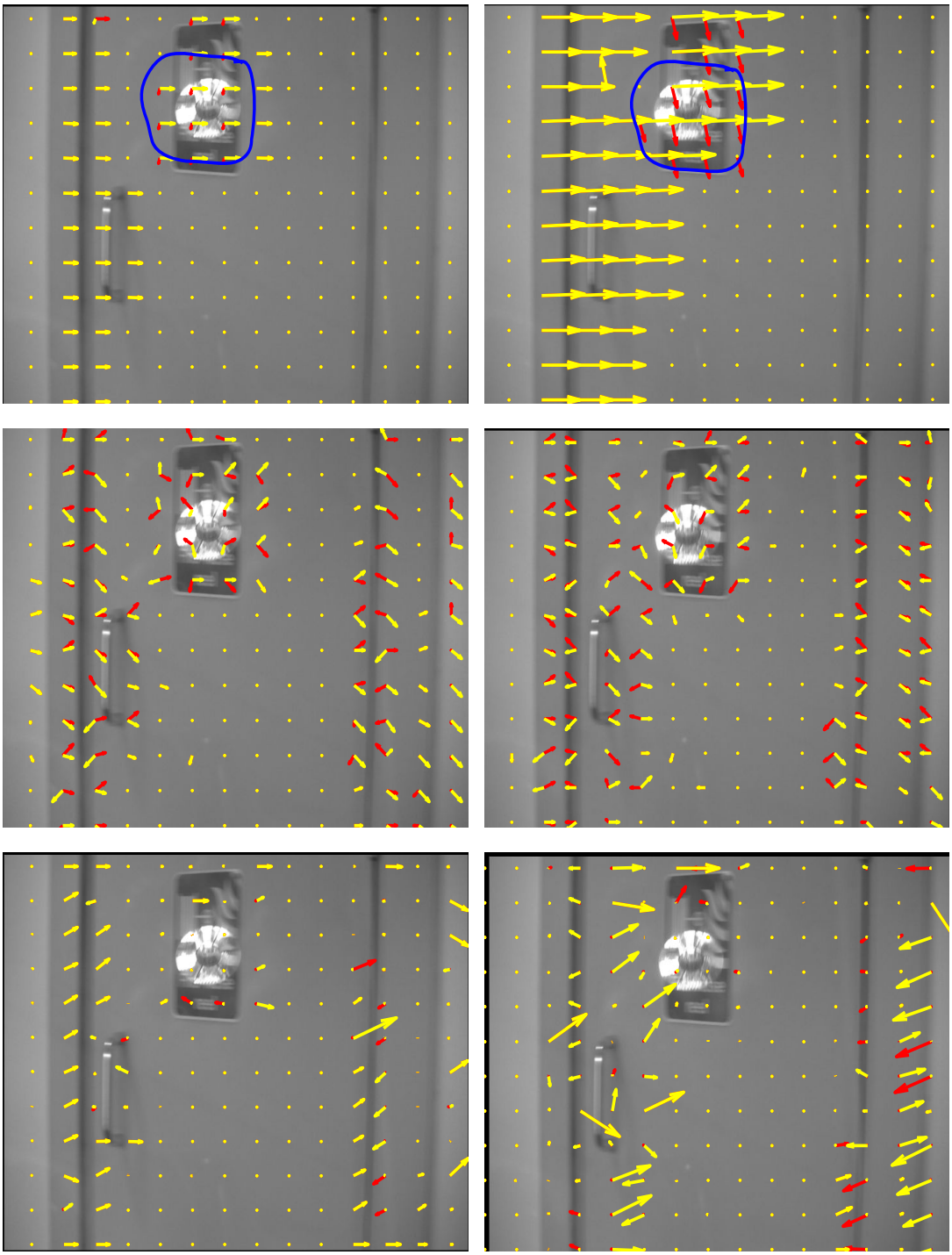


Figure 12.6: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 7 (left) and 8 (right) of* ***Bulb****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 2 for illustration clarity. Regions of interest/reflections are shown in blue.*

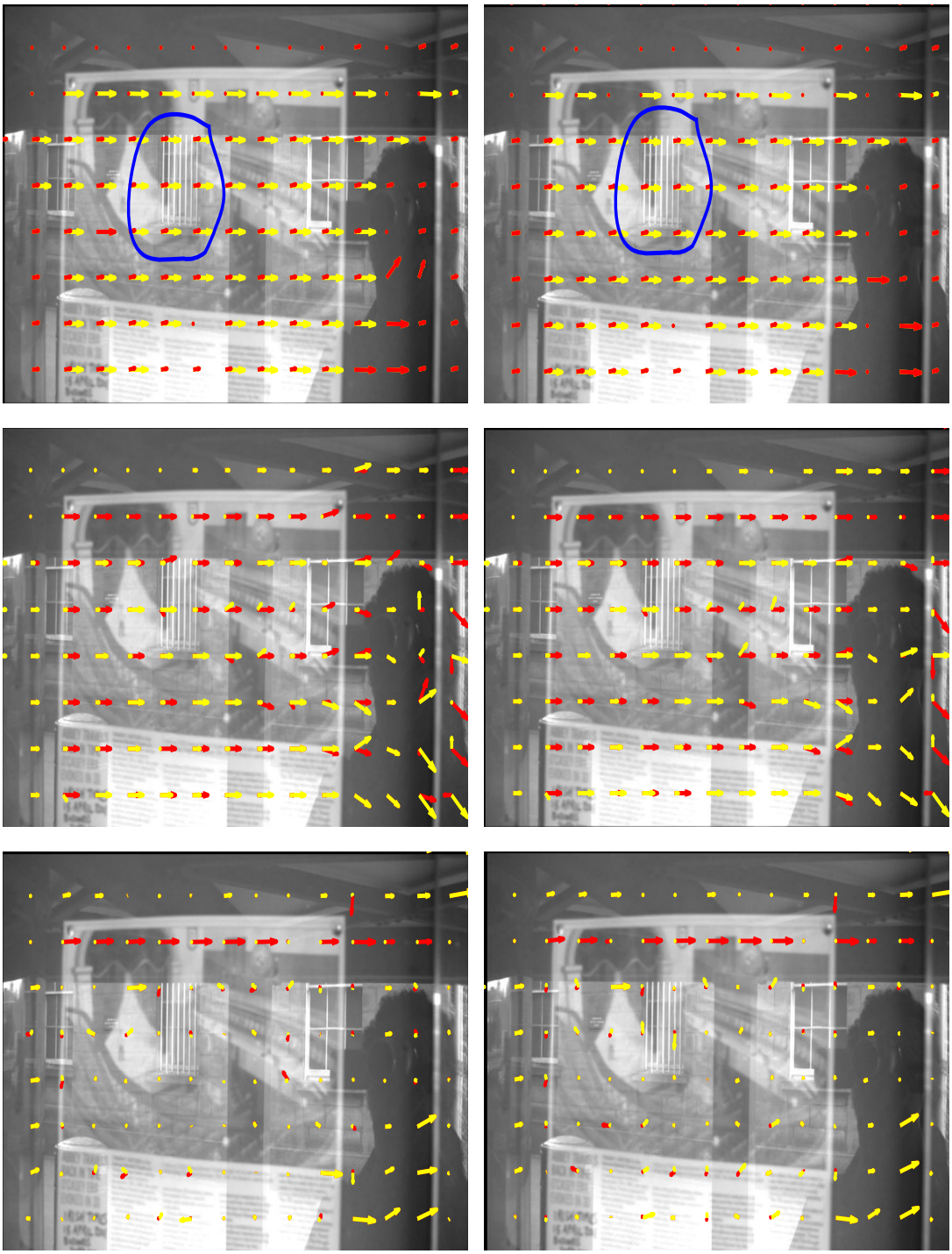


Figure 12.7: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 32 (left) and 33 (right) of* ***WindRefNoShk****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue.*

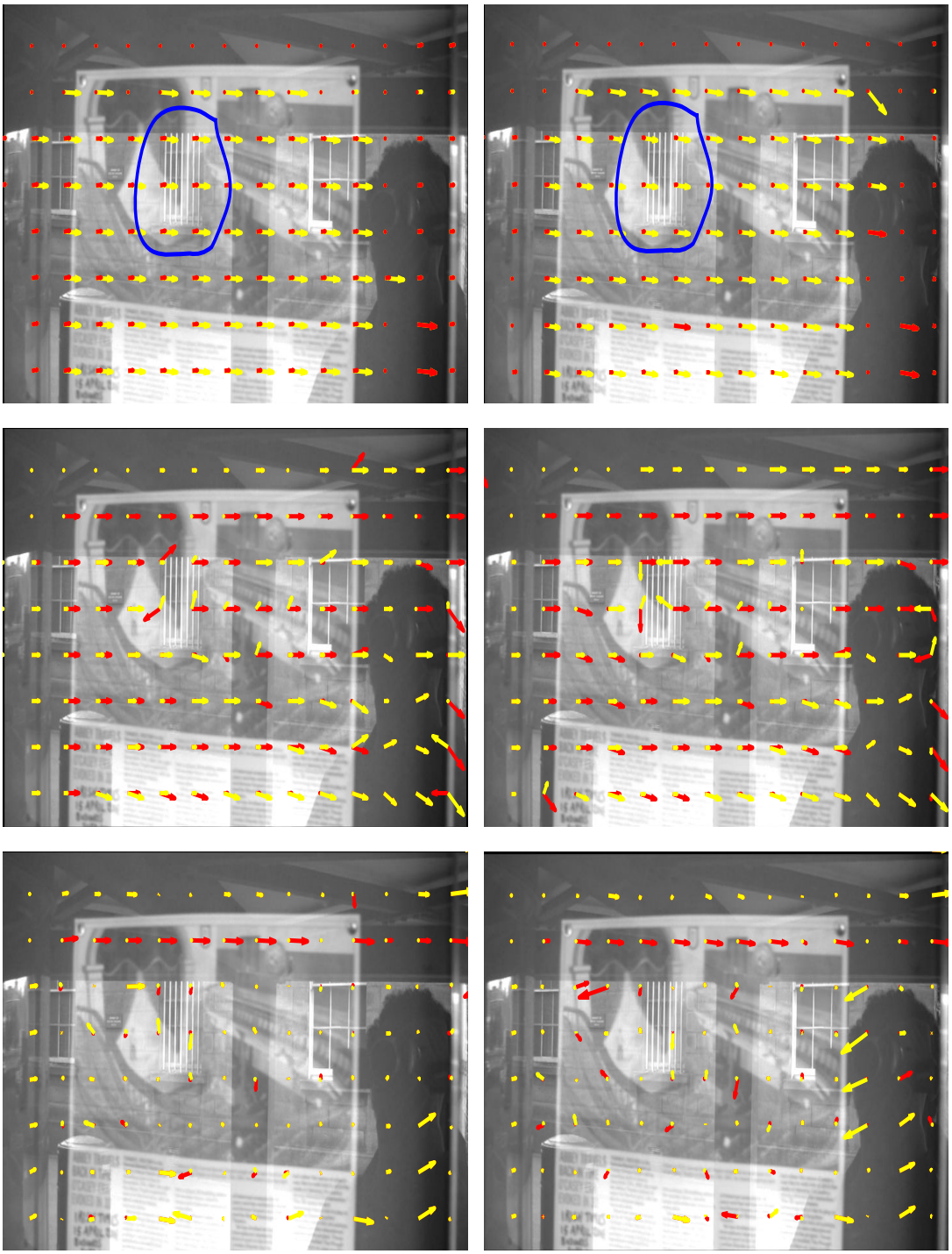


Figure 12.8: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 34 (left) and 35 (right) of* ***WindRefNoShk****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue.*

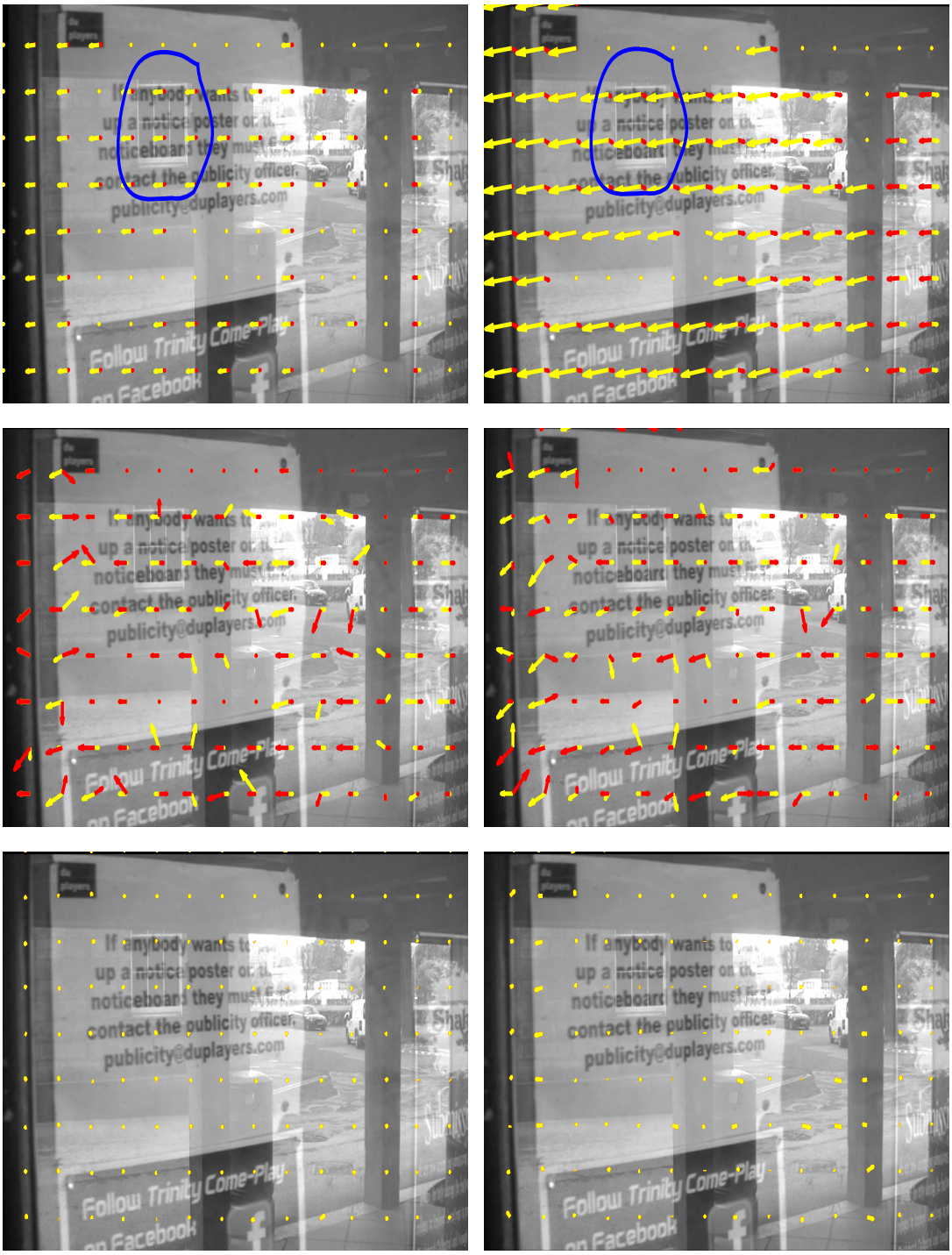


Figure 12.9: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 9 (left) and 10 (right) of* ***WindRefShk****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue.*

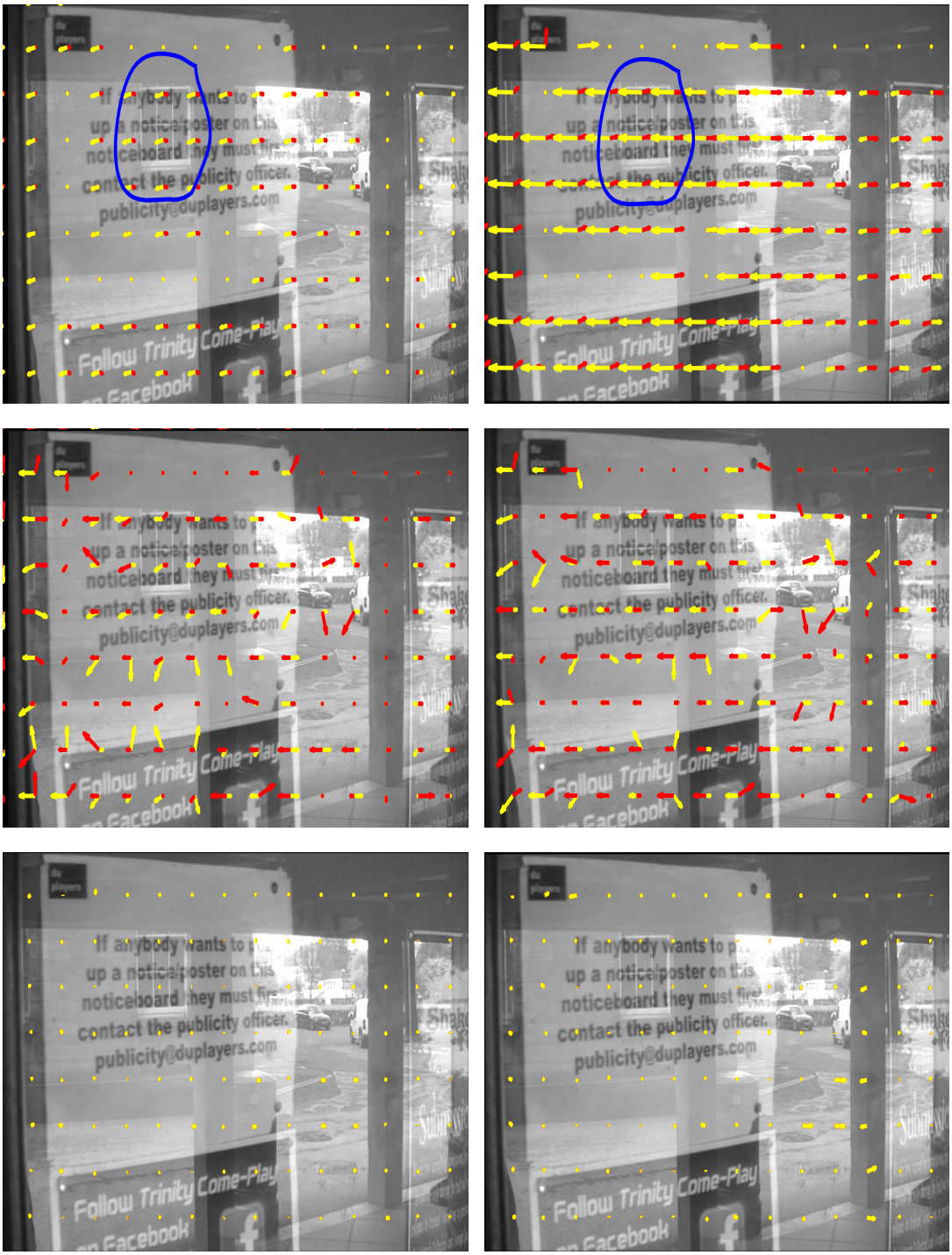


Figure 12.10: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 11 (left) and 12 (right) of* ***WindRefShk****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue.*

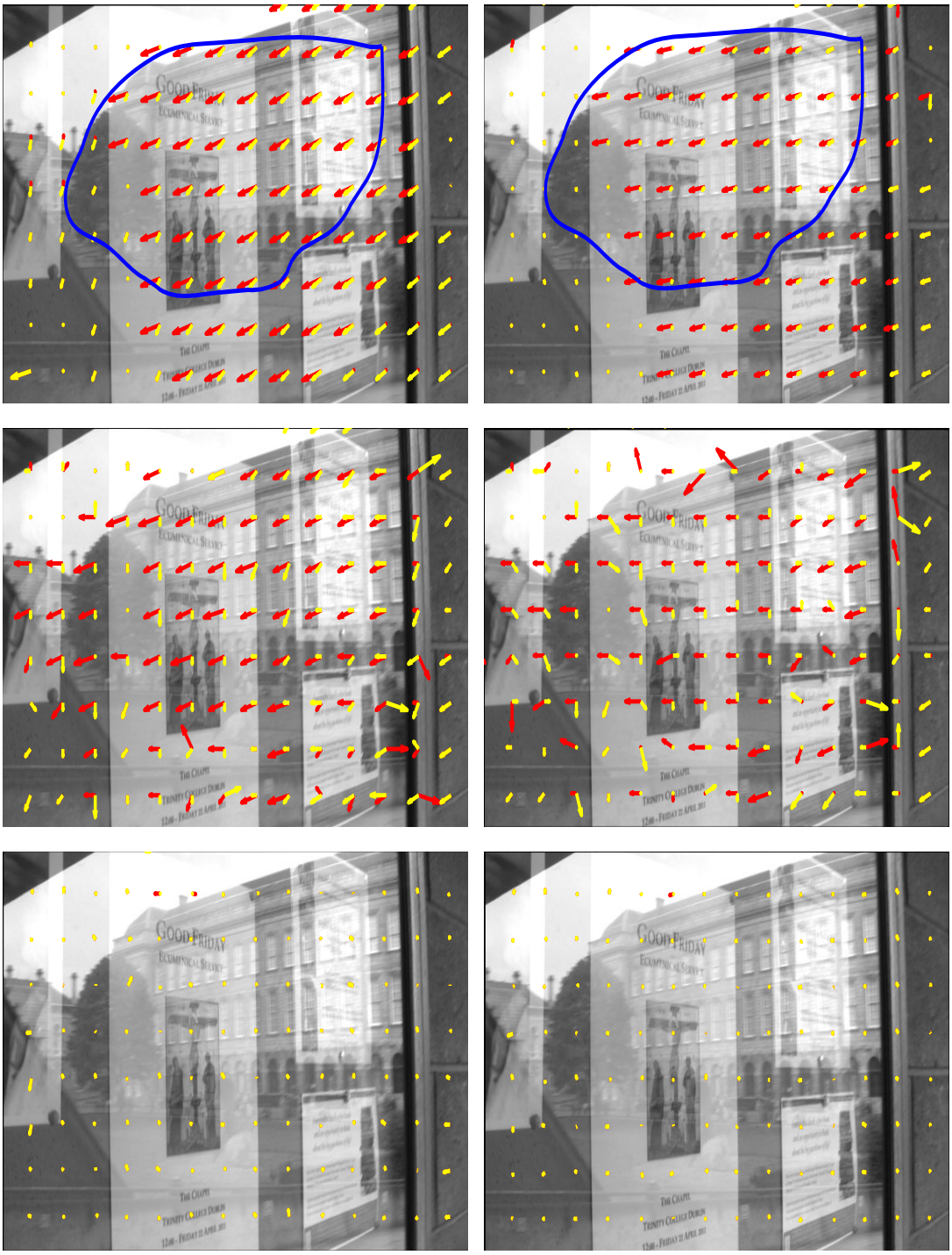


Figure 12.11: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 12 (left) and 13 (right) of* ***BuildOnWind3****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 4 for illustration clarity. Regions of interest/reflections are shown in blue.*

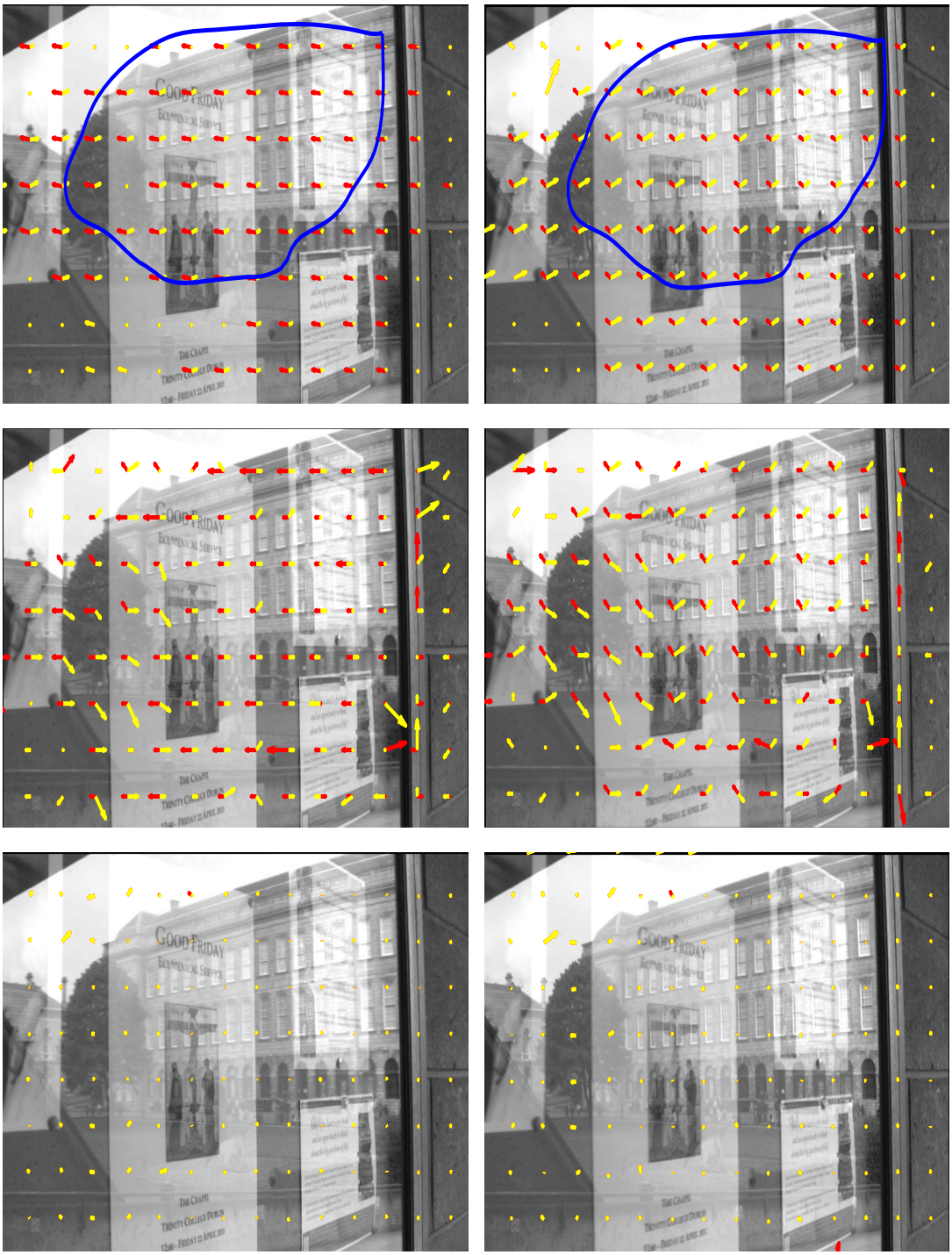


Figure 12.12: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 14 (left) and 15 (right) of* ***BuildOnWind3****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 4 for illustration clarity. Regions of interest/reflections are shown in blue.*

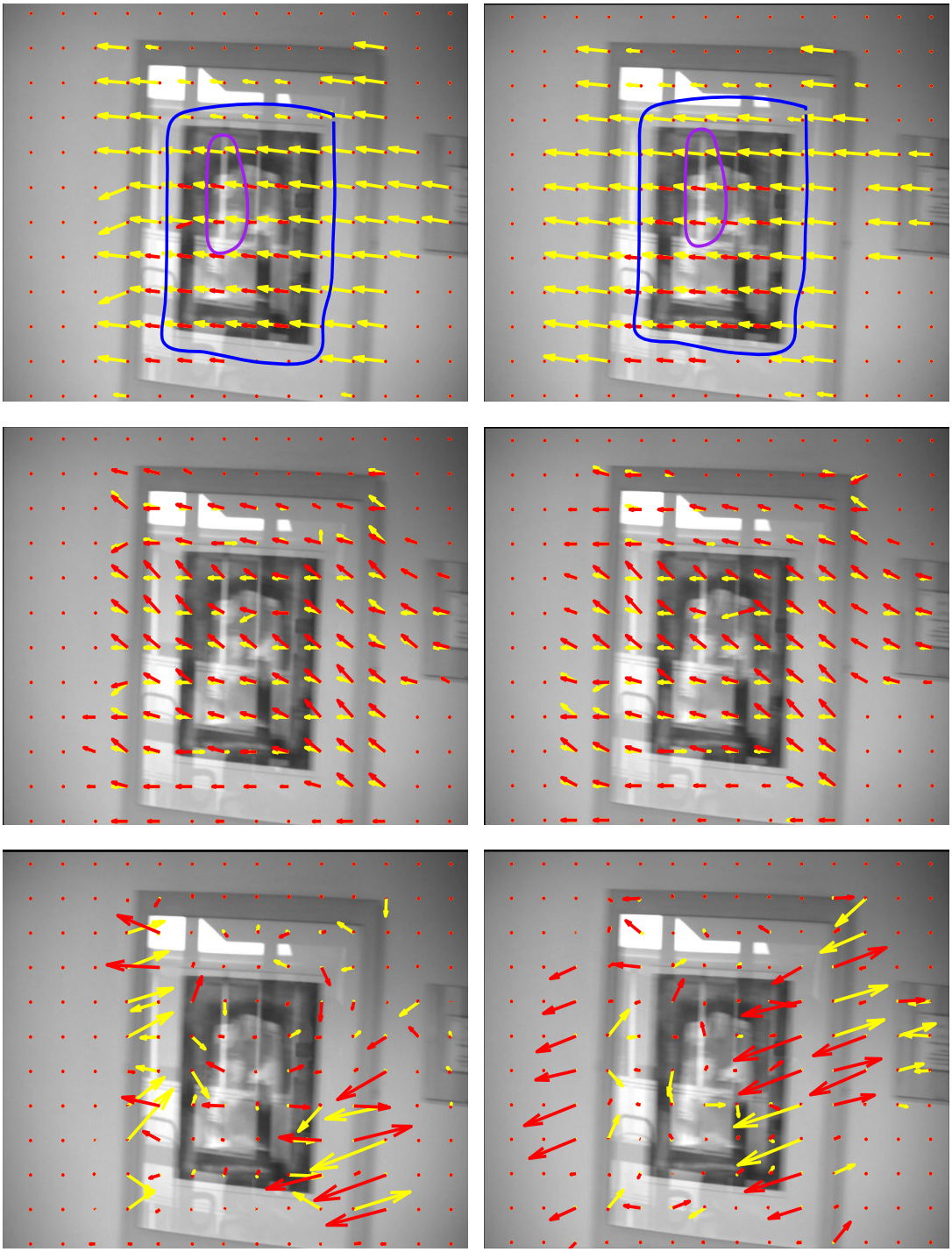


Figure 12.13: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 19 (left) and 20 (right) of* ***PortraitA2****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue. The region shown in purple undergoes an aperture effect.*

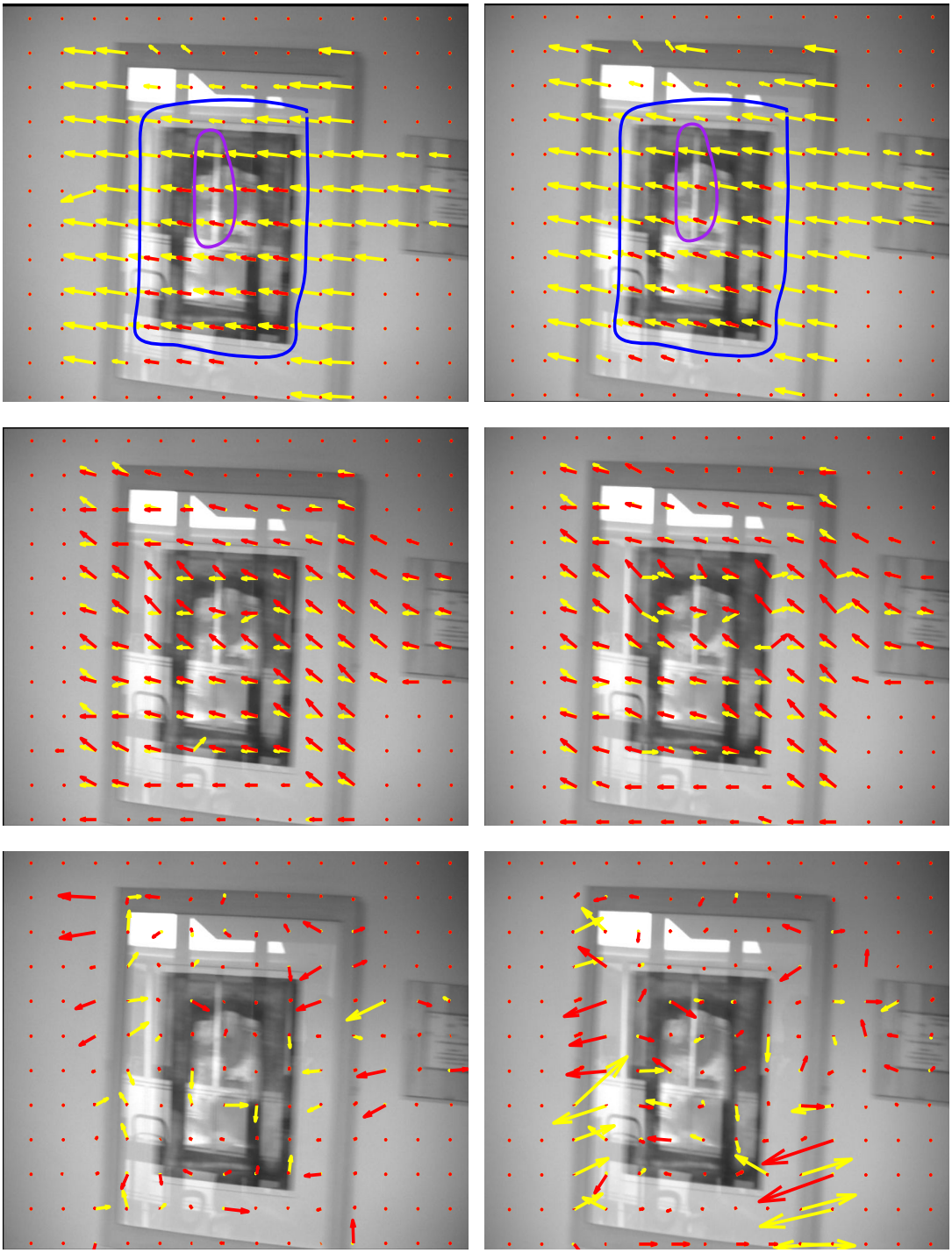


Figure 12.14: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 21 (left) and 22 (right) of* ***PortraitA2****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue. The region shown in purple undergoes an aperture effect.*

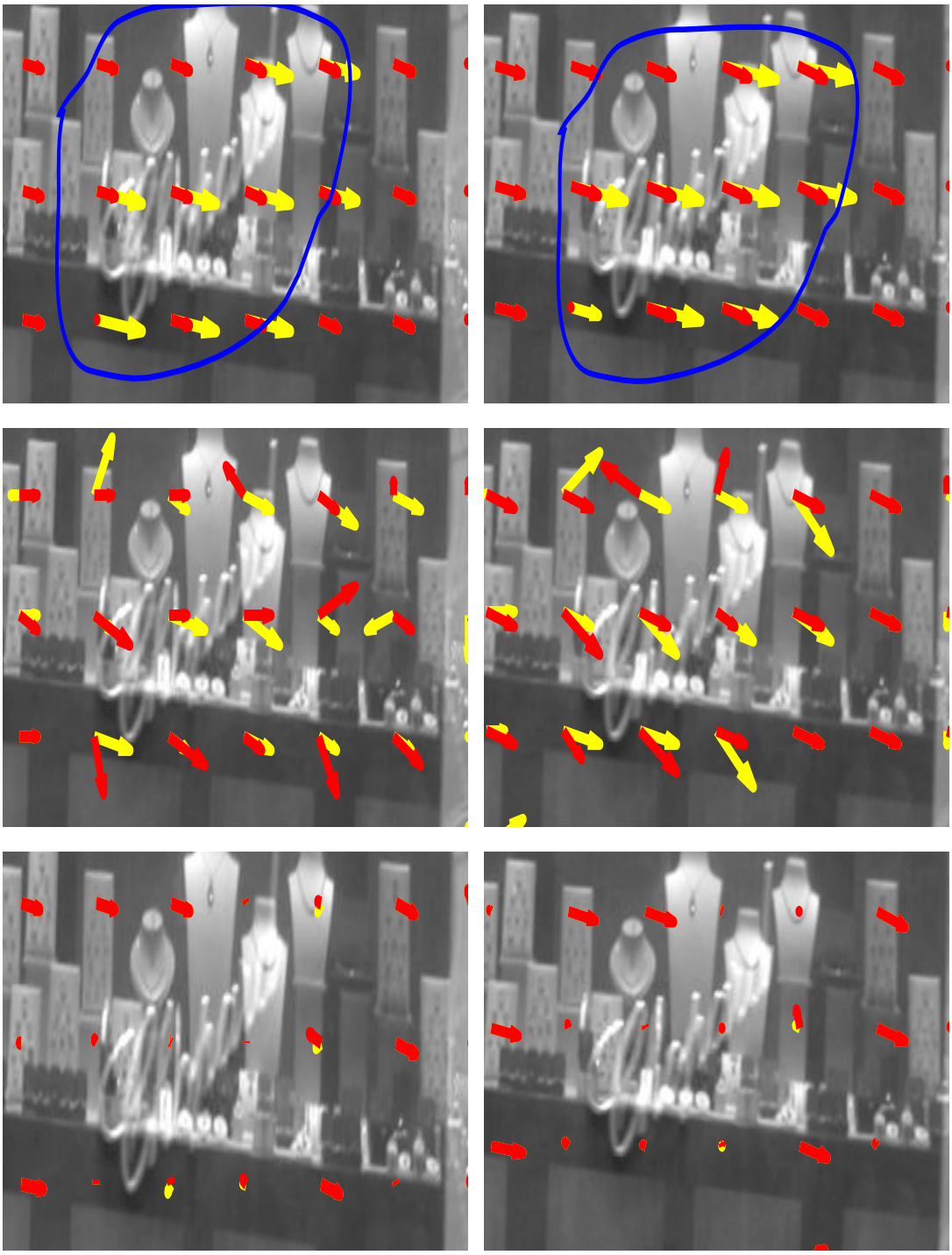


Figure 12.15: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 36 (left) and 37 (right) of* ***RedBack****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue.*

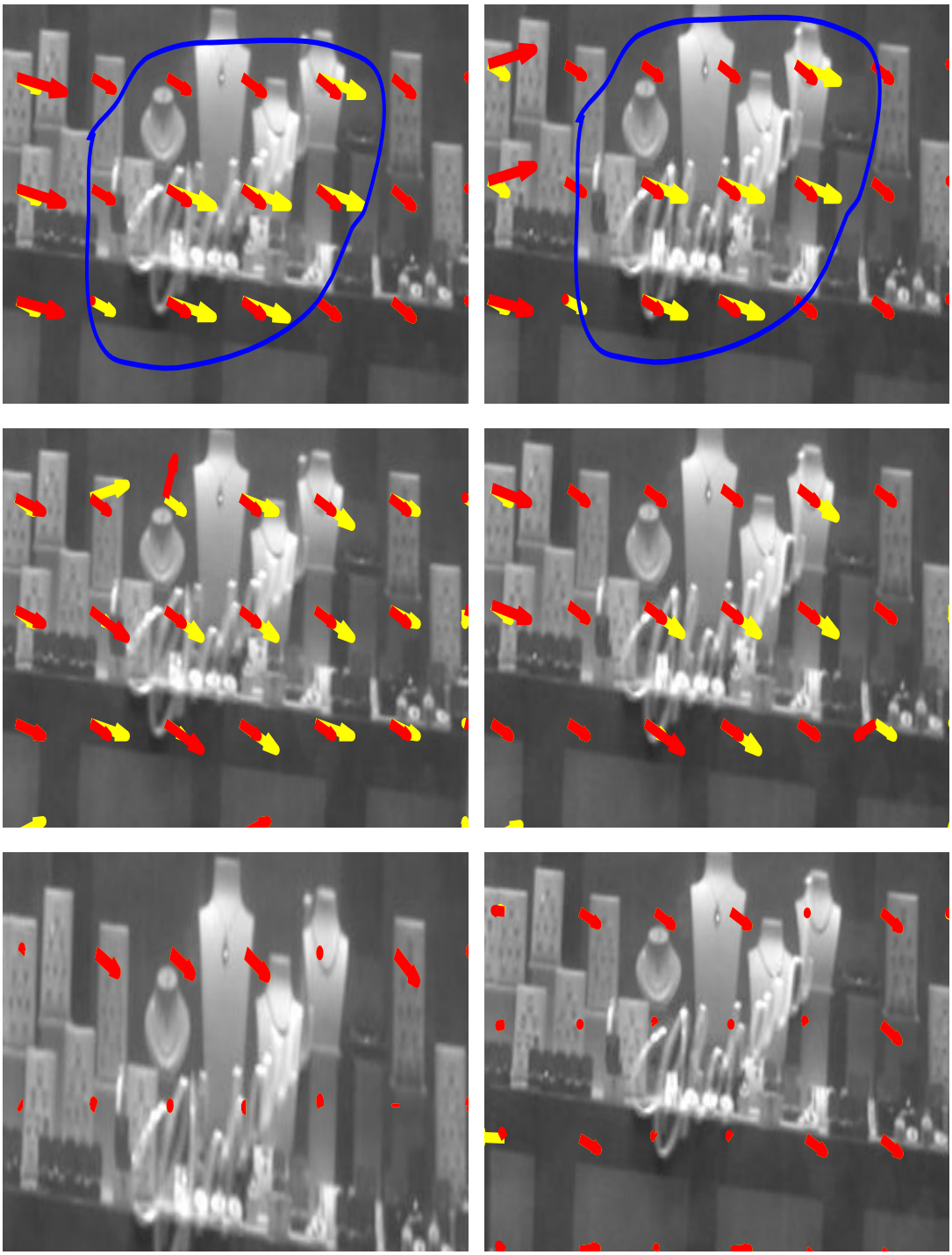


Figure 12.16: *Motion estimates generated by (from top) BIMS, FTRANS and OPTIC for frames 38 (left) and 39 (right) of* ***RedBack****. Yellow and red vectors show the background and foreground motions respectively. Vectors are scaled by a factor of 3 for illustration clarity. Regions of interest/reflections are shown in blue.*

Bibliography

[1] http://www.ces.clemson.edu/ stb/klt/.

[2] Nuke, Furnace Suite. The Foundry. www.thefoundry.co.uk.

[3] T. Aach, C. Mota, I. Stuke, M. Muhlich, and E. Barth. Analysis of Superimposed Oriented Patterns. *IEEE Transactions on Image Processing*, 15(12):3690–3700, 2006.

[4] B. Alp., P. Haavisto, and T. Jarske. Motion-preserving ranked-order filters for image sequence processing. In *SPIE Visual Communication and Image Processing*, pages 122–133, 1990.

[5] G. Arce. Multistage order statistic filters for image sequence processing. *IEEE Transactions on Signal Processing*, 39(5):1146–1163, May 1991.

[6] G. Arce and E. Malaret. Motion-preserving ranked-order filters for image sequence processing. In *International Symposium on Circuits and Systems*, pages 983–986, may 1989.

[7] V. Auvray, P. Bouthemy, and J. Linard. Motion-based segmentation of transparent layers in video sequences. In *Multimedia Content Representation, Classification and Security*, pages 298–305, 2006.

[8] X. Bai and G. Sapiro. A Geodesic Framework for Fast Interactive Image and Video Segmentation and Matting. In *International Conference on Computer Vision (ICCV)*, pages 1–8, October 2007.

[9] A. Berman, P. Vlahos, and A. Dadourian. Comprehensive method for removing from an image the background surrounding a selected object. U.S Patent 6,134,345, 2000.

[10] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image Inpainting. In *ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, pages 417–424, July 2000.

[11] R. Bornard. Probabilistic approaches for the digital restoration of television archives. PHD Thesis, Ecole Centrale Paris, 2002.

[12] A. M. Bronstein, M. M. Bronstein, M. Zibulevsky, and Y. Y. Zeevi. Sparse ICA for blind separation of transmitted and reflected images. *International Journal of Imaging Systems and Technology*, 15(1):84–91, 2005.

[13] V. Bruni, P. Ferrara, and D. Vitulano. Removal of color scratches from old motion picture films exploiting human perception. *EURASIP Journal on Advances in Signal Processing*, 2008:1–9, 2008.

[14] V. Bruni, D. Vitulano, and A. Kokaram. Fast removal of line scratches in old movies. In *International Conference on Pattern Recognition (ICPR)*, pages 827–830, 2004.

[15] R. Castagno, P. Haavisto, and G. Ramponi. A method for motion adaptive frame rate up-conversion. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(5):436 –446, 1996.

[16] N. Chen and P. De Leon. Blind image separation through kurtosis maximization. In *Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 318–322, 2001.

[17] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A Bayesian Approach to Digital Matting. In *Computer Vision and Patter Recognition (CVPR)*, volume 2, pages 264–271. IEEE Computer Society, December 2001.

[18] D. Comaniciu and P. Meer. Mean shift analysis and applications. In *International Conference on Computer Vision (ICCV)*, pages 1197–1203, 1999.

[19] D. Corrigan, N. Harte, and A. Kokaram. Pathological motion detection for robust missing data treatment. *EURASIP Journal on Advances in Signal Processing*, 2008:1–16, 2008.

[20] A. Crawford, V. Bruni, A. Kokaram, and D. Vitulano. Multi-scale semi-transparent blotch removal on archived photographs using Bayesian matting techniques and visibility Laws. In *International Conference on Image Processing (ICIP)*, pages 561–564, 2007.

[21] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 13(9):1200–1212, 2004.

[22] K. Diamantaras and T. Papadimitriou. Blind separation of reflections using the image mixtures ratio. In *International Conference on Image Processing (ICIP)*, pages 1034–1037, September 2005.

[23] A. D. Edgar. System and method for image recovery. US Patent no. 5,266,805, 1992.

[24] A. A. Efros and W. T. Freeman. Image Quilting for Texture Synthesis and Transfer. In *ACM Special Interest Group on Graphics and Interactive Techniques (SIGGRAPH)*, pages 341–346, 2001.

[25] A. A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision (ICCV)*, pages 1033–1038, 1999.

[26] H. Farid and E. Adelson. Separating reflections from images by use of Independent Components Analysis. *Journal of the Optical Society of America*, 16(9):2136–2145, 1999.

[27] R. Ferzli and L. J. Karam. A no-reference objective image sharpness metric based on the notion of just noticeable blur (JNB). *IEEE Transactions on Image Processing*, 18(4):717–728, 2009.

[28] G. Forbin, B. Besserer, J. Boldys, and D. Tschumperle. Temporal Extension to Exemplar-based Inpainting, Applied to Scratch Correction in Damaged Image Sequences. In *IASTED International Conference on Visualization, Imaging and Image Processing (VIIP)*, pages 1–5, 2005.

[29] J. Gai and S. B. Kang. Matte-based restoration of vintage video. *IEEE Transactions on Image Processing*, 18:2185–2197, October 2009.

[30] E. S. L. Gastal and M. M. Oliveira. Shared Sampling for Real-Time Alpha Matting. *Computer Graphics Forum*, 29:575–584, 2010.

[31] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann. Random walks for interactive alpha-matting. In *Visualization, Imaging, and Image Processing*, pages 423–429. ACTA Press, 2005.

[32] A. Greenblatt, K. Panetta, and S. Agaian. Restoration of semi-transparent blotches in damaged texts, manuscripts, and images through localized, logarithmic image enhancement. *International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 484–489, March 2008.

[33] Y. Guan, W. Chen, X. Liang, Z. Ding, and Q. Peng. Easy Matting - A Stroke Based Approach for Continuous Image Matting. *Computer Graphics Forum*, 25(3):567–576, 2006.

[34] T. Hang. Image Sequence Analysis. New York: Springer-Verlag, 1981.

[35] C. Harris and M. Stephens. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[36] K. He, J. Sun, and X. Tang. Fast Matting using Large Kernel Matting Laplacian. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[37] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.

[38] T. Hoshi, T. Komatsu, and T. Saito. Film blotch removal with a spatiotemporal fuzzy filter based on local image analysis of anisotropic continuity. In *International Conference on Image Processing (ICIP)*, pages 478–482, 1998.

[39] M. M. Ichir and A. Mohammad-djafari. Wavelet domain blind image separation. In *International Conference on Electronic Imaging*, volume 5207, pages 361–370, August 2003.

[40] M. M. Ichir and A. Mohammad-djafari. Hidden Markov models for Wavelet image separation and denoising. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 205–208, March 2005.

[41] K. Kayabol, E. E. Kuruoglu, and B. Sankur. Bayesian separation of images modeled with MRFs using MCMC. *IEEE Transactions on Image Processing*, 18(5):982–94, May 2009.

[42] K. Kayabol, E. E. Kuruoglu, and B. Sankur. Image source separation using color channel dependencies. In *International Conference on Independent Component Analysis and Signal Separation*, pages 499–506, March 2009.

[43] M. Kemal Gullu, O. Urhan, and S. Erturk. Scratch detection via temporal coherency analysis and removal using edge priority based interpolation. In *International Symposium on Circuits and Systems (ISCAS)*, pages 4591–4594, 2006.

[44] B. Kent, A. Kokaram, B. Collis, and S. Robinson. Two layer segmentation for handling pathological motion in degraded post production media. In *International Conference on Image Processing (ICIP)*, pages 299 – 302, 2004.

[45] P. Kisilev, M. Zibulevsky, and Y. Y. Zeevi. A multiscale framework for blind separation of linearly mixed signals. *Journal of Machine Learning Research*, 4(7-8):1339–1363, December 2003.

[46] A. Kokaram. Removal of Line Artefacts for Digital Dissemination of Archived Film and Video. In *International Conference on Multimedia Computing and Systems (ICMCS)*, pages 245–249, 1999.

[47] A. Kokaram. On missing data treatment for degraded video and film archives: a survey and a new bayesian approach. *IEEE Transactions on Image Processing*, 13(3):397–415, 2004.

[48] A. C. Kokaram. *Motion Picture Restoration*. Springer-Verlag, 1998.

[49] A. C. Kokaram and S. Godsill. MCMC for joint noise reduction and missing data treatment in degraded video. *IEEE Transactions on Image Processing , Special Issue on MCMC*, 50(2):189–205, 2002.

[50] A. C. Kokaram, R. Morris, W. Fitzgerald, and P. Rayner. Interpolation of missing data in image sequences. *IEEE Transactions on Image Processing*, 4(11):1509–1519, 1995.

[51] A. C. Kokaram and P. J. W. Rayner. System for the removal of impulsive noise in image sequences. In *SPIE Visual Communications and Image Processing*, volume 1818, pages 322–331, 1992.

[52] A. Levin, D. Lischinski, and Y. Weiss. A Closed-Form Solution to Natural Image Matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):228–242, February 2008.

[53] A. Levin and Y. Weiss. User assisted separation of reflections from a single image using a sparsity prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(9):1647–1654, September 2007.

[54] A. Levin, A. Zomet, and Y. Weiss. Separating reflections from a single image using local features. In *Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 306–313, June 2004.

[55] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.

[56] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *DARPA Image Understanding Workshop*, pages 121–130, 1981.

[57] D. M. Martinez. Model-based motion estimation and its application to restoration and interpolation of motion pictures. PHD Thesis, Mass. Inst. of Technol. Cambridge, 1986.

[58] Y. Mishima. Soft edge chroma-key generation based upon hexoctahedral color space. U.S Patent 5,355,174, 1993.

[59] C. Mota, I. Stuke, T. Aach, and E. Barth. Divide-and-Conquer Strategies for Estimating Multiple Transparent Motions. In *International Workshop on Complex Motion*, pages 66–77, 2005.

[60] M. Orchard and C. Bouman. Color quantization of images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[61] M. Pingault and D. Pellerin. Motion estimation of transparent objects in the frequency domain. *Signal Processing*, 84(4):709–719, 2004.

[62] R. J. Qian and M. I. Sezan. Video Background Replacement Without a Blue Screen. In *International Conference on Image Processing (ICIP)*, pages 143–146, 1999.

[63] A. Rares, M. Reinders, and J. Biemond. Complex event classification in degraded image sequences. In *International Conference on Image Processing (ICIP)*, pages 253–256, 2001.

[64] C. Rhemann, C. Rother, and M. Gelautz. Improving Color Modeling for Alpha Matting. In *British Machine Vision Conference (BMVC)*, pages 1155–1164, 2008.

[65] C. Rhemann, C. Rother, A. Rav-Acha, and T. Sharp. High resolution matting via interactive trimap segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1 –8, 2008.

[66] C. Rhemann, Christophand Rother, P. Kohli, and M. Gelautz. A Spatially Varying PSF-based Prior for Alpha Matting. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2149–2156, 2010.

[67] D. Ring and F. Pitie. Feature-Assisted Sparse to Dense Motion Estimation Using Geodesic Distances. In *Irish Machine Vision and Image Processing*, pages 7–12, 2009.

[68] V. Roosmalen. Restoration of archived film and video. PHD Thesis, Delft University, 1999.

[69] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[70] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Computer Vision and Pattern Recognition (CVPR)*, pages 18–25, 2000.

[71] H. Sahlin and H. Broman. Blind separation of images. In *Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 109–113, November 1996.

[72] T. Saito, T. Komatsu, T. Ohuchi, and T. Hoshi. Practical nonlinear filtering for removal of blotches from old film . In *International Conference on Image Processing (ICIP)*, pages 164–168, 1999.

[73] B. Sarel and M. Irani. Separating transparent layers through layer information exchange. In *European Conference on Computer Vision (ECCV)*, pages 328–341, May 2004.

[74] B. Sarel and M. Irani. Separating Transparent Layers of Repetitive Dynamic Behaviors. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 26–32, October 2005.

[75] M. Shizawa and K. Maze. Unified computational theory for motion transparency and motion boundaries based on Eigenenergy analysis. In *Computer Vision and Pattern Recognition (CVPR)*, pages 289 –295, 1991.

[76] A. R. Smith and J. F. Blinn. Blue screen matting. In *Computer graphics and interactive techniques*, pages 259–268, 1996.

[77] W. Souidene, A. Aissa-El-Bey, K. Abed-Meraim, and A. Beghdadi. Blind image separation using sparse representation. In *International Conference on Image Processing (ICIP)*, volume 3, pages 125–128, October 2007.

[78] F. Stanco, G. Ramponi, and L. Tenze. Removal of Semi-Transparent Blotches in Old Photographic Prints. In *European Cooperation in Science and Technology (COST) 276 Workshop*, pages 1–5, 2003.

[79] F. Stanco, L. Tenze, and A. De Rosa. An improved method for water blotches detection and restoration. In *International Symposium on Signal Processing and Information Technology*, pages 457–460, 2004.

[80] R. Storey. Electronic detection and concealment of film dirt. UK Patent Specification no. 2139039, 1984.

[81] I. Stuke, T. Aach, E. Barth, and C. Mota. Multiple-Motion-Estimation by Block-matching using MRF. *International Journal of Computer and Information Science*, 5(1), 2004.

[82] J. Sun, J. Jia, C. keung Tang, and H. yeung Shum. Poisson matting. *ACM Transactions on Graphics*, 23:315–321, 2004.

[83] R. Szeliski, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *Computer Vision and Pattern Recognition (CVPR)*, pages 246–253, 2000.

[84] D. Tegolo and F. Isgro. Scratch detection and removal from static images using simple statistics and genetic algorithms. In *International Conference on Image Processing (ICIP)*, 2001.

[85] C. Tomasi and T. Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.

[86] J. Toro, F. Owens, and R. Medina. Multiple motion estimation and segmentation in transparency. In *Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2087–2090, 2000.

[87] J. Toro, F. Owens, and R. Medina. Multiple motion estimation and segmentation in transparency. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2087–2090, 2000.

[88] D. Vernon. Decoupling Fourier components of dynamic image sequences: a theory of signal separation, image segmentation and optical flow estimation. In *European Conference on Computer Vision (ECCV)*, pages 68–85, 1998.

[89] P. Viola and M. Jones. Robust Real-time Object Detection. In *International Workshop on Statistical and Computational Theories of Vision-Modeling, Learning, Computing, and Sampling*, pages 1–25, 2001.

[90] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 936–943, October 2005.

[91] J. Wang and M. Cohen. Optimized color sampling for robust matting. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[92] J. Wang, N. Patel, and W. Grosky. Video frame rate up conversion using region based motion compensation. In *Electro/Information Technology Conference*, pages 143–157, 2004.

[93] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[94] Y. Weiss. Deriving intrinsic images from image sequences. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 68–75, July 2001.

[95] M. Yamazaki, Y.-W. Chen, and G. Xu. Separating reflections from images using kernel Independent Component Analysis. In *International Conference on Pattern Recognition (ICPR)*, volume 3, pages 194–197, August 2006.

[96] Y. Zheng, C. Kambhamettu, Y. Zheng, and C. Kambhamettu. Learning Based Digital Matting. In *International Conference on Computer Vision (ICCV)*, pages 889–896, 2009.

[97] M. Zibulevsky, Y. Y. Zeevi, P. Kisilev, and B. Pearlmutter. Blind source separation via multinode sparse representation. In *Neural Information Processing Systems 12*, pages 1049–1056, 2001.

[98] S. zu Hou, M. Wang, X. cheng Tian, X. Wang, and L. yan Huang. Research of image separation based on improved Independent Component Analysis. In *International Conference on Signal Processing*, volume 2, pages 16–20, November 2006.