USING ONE GRAPH-CUT TO FUSE MULTIPLE CANDIDATE MAPS IN DEPTH ESTIMATION

F. Pitié

Sigmedia Group, Trinity College Dublin, fpitie@mee.tcd.ie

Abstract

Graph-cut techniques for depth and disparity estimations are known to be powerful but also slow. We propose a graph-cut framework that is able to estimate depth maps from a set of candidate values. By employing a restricted set of candidates for each pixel, rough depth maps can be effectively refined to be accurate, smooth and continuous. The contribution of this work is to extend the graph structure proposed in the original papers on Graph-Cuts by Ishikawa and Roy, in such a way that sparse sets of candidates can be handled in one graph-cut.

Keywords: Graph-Cut, Depth Map Estimation,

1 Introduction

In 3D reconstruction from stereo, the problem of depth estimation can be thought of as a labelling problem. A convenient way of dealing with the labelling is to employ a Bayesian framework and consider that the depth map $\mathcal{D}$ can be modeled as a Markov Random Field (MRF). The interest of the Bayesian framework is that prior information on the spatial smoothness of the depth map can also be included. Typically, the energy of the MRF to be minimized will have the following form:

$$E(\mathcal{D}) = \sum_p U_p(d_p) + \sum_{p,q} V_{p,q}(d_p, d_q) \quad (1)$$

The function $U_p(d_p)$ is linked to the likelihood of having depth d_p at pixel p . A small value $U_p(d_p) \approx 0$ means that this depth is very likely. The function $V_{p,q}(d_p, d_q)$ is a smoothness penalty function, which essentially penalizes large difference of depth between neighbouring pixels.

This kind of energy has received a great deal of attention in recent years. Numerous optimization schemes have been proposed such as Simulated Annealing, Iterated Condition Mode, Belief Propagation, Tree Reweighted Message Passing or Graph-Cuts. In this paper we are interested in Graph-Cuts, a technique that has been used for some time now to solve smoothness constraints in labelling problems. The work of Ishikawa [2] and Roy [7, 6] brought graph cut to the attention of the computer vision community. The idea is to encode the likelihood and smoothness terms inside the capacities of a graph edges. The labelling is solved by finding a minimum cut through the graph. Each pixel is labelled as 0 or 1, depending on each side of the cut it lies. Graph-cuts have

been very successful at solving hard problems. They come however with some limitations since, in the general case, they can only solve the 2-labels problem. For n-labels, the general problem is believed to be NP hard and approximations to the exact solution must be considered. The works of Boykov, Kolmogorov and others have shown that with techniques such as alpha-expansion [1] and QPBO [5], n-labeling problems could be efficiently decomposed into a succession of 2-label problems.

A remarkable point in the original work of Ishikawa and Roy, is that an exact solution to a multi-labelling problem is however possible, with the caveat that the smoothness of the label field should be convex. For instance, the smoothness can follow the L_1 norm:

$$V_{p,q}(d_p, d_q) = \alpha_{p,q} |d_p - d_q| \quad (2)$$

where $\alpha_{p,q}$ is a texture parameter, independent of the depth values. With this kind of penalty, Ishikawa [2] and Roy [7, 6] showed that it is possible to solve the multi-labelling by means of a minimum cut on a graph. The graph is of dimension width x height x number of labels. The depth likelihood and the spatial smoothness are encoded in the capacities of the graph edges. The labels can be extracted by cutting the graph, and reporting the edges that have been cut. Similar ideas have also been reported in more recent work such as Paris *et al.* [3] and it is also closely related to Total Variation [4], which offers a continuous framework to this problem.

Note that the convexity constraint on the smoothness terms has some important consequences. One issue is that solutions tend to minimize depth discontinuities. Thus large depth discontinuities can be over-smoothed. This can be mitigated by playing with the texture parameter $\alpha_{p,q}$, which can break the smoothness at image edges.

One practical problem with graph-cut techniques is that they can take a considerable amount of time and memory to label large pictures. With current implementations of graph-cut, such as Push-Relabel, the complexity min-cut for a graph with E edges and V vertices is $O(EV^2)$. Thus the computation load increases with the cube of the number of pixels in the picture. For a small 300x300 picture, with 50 possible depth values, Ishikawa and Roy's graph will already contain 4.5 million vertices. The memory will quickly be filled up by the sheer size of the graph and the estimation might take a few minutes. The computational aspect is thus a real issue when using this graph-cut technique in practice and ways must be found to reduce the volume of information.

Contribution. The contribution of this paper is to provide a mechanism for solving Ishikawa and Roy’s graph using a sparse set of candidates. We argue that in most cases it is possible to obtain pretty good depth maps using fast but sub-optimal techniques, such as dynamic programming. Thus starting from a rough estimate, it is possible to propose for each pixel a set of a half dozen depth candidate that will contain the true depth value. Then, instead of working the full volume of disparities, we actually only need to consider a more manageable set of candidate values per pixel.

Using candidates in depth estimation is not a new idea. Woodford *et al.* [8] showed that depth maps can be fused using QPBO [5], an adaptation of the 2-label graph-cut technique. QPBO can only fuse two label fields; thus a number of iterations are required to refine the depth map. The advantage is that more complex smoothness model can be used and that the memory usage remains compact. The problem is the iterative nature of the method which leads to high computational times and convergence issues. We propose instead to start from the graph structure of Ishikawa and Roy’s graph and solve for multiple candidates at once and still obtain a global minimum. Our contribution is to provide a mechanism for setting up the weights in the reduced graph, in such way that the cut still yields a L^1 norm smoothness between candidate values.

Organisation of the Paper. In the next section we describe Ishikawa and Roy’s graph structure. In section 3 we describe how the graph should be modified to accommodate candidate depths. The validity of the method is discussed in section 4 and practical results for depth estimation are presented in section 5.

2 Ishikawa’s and Roy’s Graph

The graph structure presented in Ishikawa’s and Roy’s work is displayed in Fig. 1. The x and y axis of the graph correspond to the coordinates of the picture and the d axis corresponds to the depth. Each **edge** of the graph corresponds one of the terms of the energy of equation (1). For the x -edges and y -edges, the capacity is given by the texture parameter $\alpha_{p,q}$ between neighbouring pixels $p = (x, y)$ and $q = (x', y')$. For the d -edges, the capacity is $U_p(d)$, the penalty for having d at pixel (x, y) . Thus for a particular node $p = (x, y, d)$, the capacities of the incoming edges are given by:

$$(x, y, d) \leftrightarrow (x, y, d+1) = U_{(x,y)}(d+1) \quad (3)$$

$$(x, y, d) \leftrightarrow (x+1, y, d) = \alpha_{p,q} \quad (4)$$

$$(x, y, d) \leftrightarrow (x, y+1, d) = \alpha_{p,q} \quad (5)$$

For each pixel, the highest node (minimum depth) of the column is linked to a common special node (S), referred to as the source, and the lowest node (maximum depth) is linked to another special node (T), referred to as the sink:

$$(x, y, d_{\min}) \leftrightarrow S = U_{(x,y)}(d = d_{\min}) \quad (6)$$

$$(x, y, d_{\max} - 1) \leftrightarrow T = U_{(x,y)}(d = d_{\max}) \quad (7)$$

A $S - T$ cut of the graph separates the graph into two parts:

Figure 1: Regular lattice for the global graph cut minimisation. On the left, the 3D lattice of the graph. On the right, a 2D slice. The likelihood terms are reported on the vertical edges and the spatial smoothness penalty (x direction) is encoded in the horizontal edges. The minimum $s - t$ cut gives the Maximum A Posteriori depth map, eg. in this case $d_A = 2$, $d_B = 3$ and $d_C = 2$.

one that includes the source S and one that includes the sink T. The sum of all edges that have been cut is actually the overall energy $E(\mathcal{D})$ for the depth map. The depth map can be found by performing a *minimum graph cut* on this huge graph and reporting all the edges that have been cut as the depth for each pixel. For instance, after graph cut in Figure 1(b), the depth map for the three pixels is $d_A = 2$, $d_B = 3$ and $d_C = 2$.

To see that the L^1 smoothness is induced, consider that after graph cut the depth is $d_A = 0$ and $d_B = 10$. Then the 10 horizontal edges that are between A_1 and B_{10} are cut. All these edges have constant weight $\alpha_{A,B}$, thus the cut will have indeed a total cost of $10 \alpha_{A,B} = \alpha_{A,B} |d_B - d_A|$.

3 Candidate Disparities

If finding the minimum cut is quite fast for small graphs, the method becomes unpractical for “very large” graphs. Typically, for image resolutions bigger than 300x300 pixels and a depth range greater than 60 values, graph cut will take a few minutes on modern CPU’s. Working with a subset of *candidate* disparities is thus an interesting alternative. Also, candidate depth values do not have to take integer values, thus by working with candidates we can obtain continuous depth maps.

In previous section graph structure, the L^1 penalty is induced by the “height” of each depth and thus the number of x and y edges that are cut. A similar mechanism can be used when dealing with candidates, albeit the weights for the x and y axis will not be constant but will depend on the candidate values.

A two pixels example of a possible graph is given in Fig. 2. Pixel A has 5 depth candidates, denoted as $\{d_A(1), \dots, d_A(5)\}$. Pixel B has 4 depth candidates,

Figure 2: Example of a graph using candidate selection.

denoted as $\{d_B(1), \dots, d_B(4)\}$. Using the same setup as in Ishikawa and Roy, the graph comprises of $5-1=4$ nodes for the pixel column A and $4-1=3$ nodes for the pixel column B . Similarly the vertical edges correspond to the depth likelihood terms, eg. the edge (A_1, A_2) has a capacity of $U_A(d_A(2))$. The weights for the edges between A and B have been set using the algorithm proposed hereafter. It is easy to verify that any cut will be proportional to the absolute difference of depths. Take for instance a graph cut (in red), which goes through $d_A(2)$ and $d_B(4)$. The total cost for cutting the x and y edges is $2.5 + 0.5 + 1.5 + 2 + 1 = 7.5$, which is also the depth difference $d_B(4) - d_A(2) = 16 - 8.5$.

We propose in the following an algorithm that will automatically set these weights on the links between the A and B 's nodes.

3.1 Graph Construction.

It is sufficient to know how to connect two interacting pixels, since the same principle is then applied to all interacting pixel pairs. The construction of the graph is done in an iterative way. Consider the case of two pixels A and B with depth candidates $\{d_A(i)\}_{i \leq N_A}$ and $\{d_B(j)\}_{j \leq N_B}$. To simplify the notations, we define that $A_0 = S = B_0$ and $A_{N_A} = B_{N_B} = T$. Also we note $d_A(i > N_A) = +\infty$ and $d_B(j > N_B) = +\infty$.

Consider that after some iterations, $A_1, \dots, A_i$ and $B_1, \dots, B_j$ have been connected (see Fig. 3). Denote

$$d' = \min(d_A(i+1), d_B(j+1)). \quad (8)$$

If $d' = d_A(i+1) \leq d_B(j+1)$, then we add the edge (A_{i+1}, B_j) to the graph and we give it the capacity of

$$w = \alpha_{A,B} (\min(d_A(i+2), d_B(j+1)) - d_A(i+1)) \quad (9)$$

Conversely, if $d' = d_B(j+1) < d_A(i+1)$, then we add the

Algorithm 1 Setting up the smoothness edges for candidate disparities. Using notations $B(0) \equiv A(0) \equiv S$, $B(N_B) \equiv A(N_A) \equiv T$ and $d_A(i > N_A) = +\infty$, $d_B(i > N_B) = +\infty$.

$i \leftarrow 0; j \leftarrow 0$

while $(i < N_A)$ **or** $(j < N_B)$ **do**

if $d_A(i+1) \leq d_B(j+1)$ **then**

Connect A_{i+1} to B_j with edge capacity

$c = \alpha_{A,B} [\min(d_A(i+2), d_B(j+1)) - d_A(i+1)];$

$i \leftarrow i+1;$

else

Connect B_{j+1} to A_i with edge capacity

$c = \alpha_{A,B} [\min(d_A(i+1), d_B(j+2)) - d_B(j+1)];$

$j \leftarrow j+1;$

end if

end while

edge (A_i, B_{j+1}) to the graph and we give it the capacity of

$$w = \alpha_{A,B} (\min(d_A(i+1), d_B(j+2)) - d_B(j+1)) \quad (10)$$

This process is summarized in Alg. 1.

3.2 Proof of Correctness

We need to show that any cut that goes through an edge (A_p, A_{p+1}) and (B_q, B_{q+1}) will have a cost of the form $C = \alpha_{A,B} |d_A(p+1) - d_B(q+1)|$.

To prove this we can proceed by recurrence. For two pixels A and B with disparities $\{d_A(1), \dots, d_A(N_A)\}$ and $\{d_B(1), \dots, d_B(N_B)\}$, we assume that all the edges have been set, up to A_i and B_j .

Let $d' = \min(d_A(i+1), d_B(j+1))$. As illustrated in Fig. 4, consider a cut that goes through the edge (A_i, B_j) and exists through any other edge (A_p, A_{p+1}) or (B_q, B_{q+1}) with depth d . Then the recurrence hypothesis is that the cost of cutting all the horizontal links between $\{A_1, \dots, A_i\}$ and $\{B_1, \dots, B_j\}$ is $C = \alpha_{A,B} |d' - d|$. For instance in Fig. 4 (where $\alpha_{A,B} = 1$), the orange cut that leaves the sub-graph at (S, B_1) is assumed to have a combined cost of $d' - d_B(1)$. Note that the edge capacity of (S, B_1) is a likelihood edge, and is not included in the horizontal edges.

If $i = 0$ and $j = 0$, then $|d_A(0) - d_B(0)| = 0$ and the recurrence hypothesis is satisfied.

Without loss of generality, assume that $d_A(i+1) < d_B(j+1)$ as in Fig. 3. Then, $d' = d_A(i+1)$ and by construction, A_{i+1} is connected to B_j . Denote $d'' = \min(d_A(i+2), d_B(j))$. The capacity of the new edge is by construction $w = \alpha_{A,B} (d'' - d') = \alpha_{A,B} (d'' - d_A(i+1))$. Consider now the recurrence hypothesis. A cut that goes through (A_{i+1}, B_j) can either leaves the subgraph by the nearby edge (A_i, A_{i+1}) or by an outer edge of the previously built subgraph that corresponds to a depth d (see Fig. 3). In the first case, the cost of the cut will be only $w = \alpha_{A,B} (d'' - d_A(i+1))$, which is consistent with the recurrence hypothesis. In the latter case, the cut will go through (A_i, B_j) and we can thus use the recurrence hypothesis for (A_i, B_j) .

Figure 3: Recurrence Steps. The graph has been connected up to the situation on the left. Since $d_A(i+1) \leq d_B(j+1)$, we connect A_{i+1} to B_j . On the right, we can see that a cut going through (A_{i+1}, B_j) either leaves from (A_i, A_{i+1}) or from an edge higher than (A_i, B_j) , in which case the Recurrence Hypothesis holds.

The cost is then $w + \alpha_{A,B}(d' - d) = \alpha_{A,B}(d'' - d' + d' - d) = \alpha_{A,B}(d'' - d)$. QEF.

4 Application to Depth Estimation Refinement

To show the interest of the technique, we start from a rough depth map, and refine the map by applying the global energy of equation (1) using depth candidates coming from the rough map. Here the initial depth map is estimated using Dynamic Programming (DP) along the scan line and using the normalized cross correlation over 3x3 blocks to build the likelihood. An example of such depth map is given in Fig. 5-(c). Since the optimization is done independently along scanlines, the resulting depth map suffers from heavy jitter artefacts.

We thus apply the model of equation (1) for a set of depth candidates generated from this initial depth map. In the first candidate pool, we consider the initial depth value for each pixel and add some of the neighboring depth values as candidates. The neighbouring candidates are taken as being the 8 most frequent depth values in a surrounding 50x50 block:

$$C_1 = \{d_{(x,y)}, d_F^1(x,y), d_F^2(x,y), d_F^3(x,y), d_F^4(x,y), d_F^5(x,y), d_F^6(x,y), d_F^7(x,y), d_F^8(x,y)\} \quad (11)$$

At the second iteration, the depth map is further refined by considering again some of the newly most frequent disparities plus extra variations:

$$C_2 = \{d_{(x,y)}, d_F^1(x,y), d_F^2(x,y), d_F^3(x,y), d_{(x,y)} + 1, d_{(x,y)} + 2, d_{(x,y)} - 1, d_{(x,y)} - 2\} \quad (12)$$

It is now easy to achieve a continuous map by iteratively

Figure 4: Recurrence Hypothesis. The sub-graph (solid lines) has been constructed up to nodes A_i and B_j . Let $d' = \min(d_A(i+1), d_B(j+1))$. Consider a cut that goes through the edge (A_i, B_j) and exists through any other edge (A_p, A_{p+1}) or (B_q, B_{q+1}) with depth d . Then the recurrence hypothesis is that the cost of cutting all the links between $\{A_1, \dots, A_i\}$ and $\{B_1, \dots, B_j\}$ is $C = |d' - d|$. For instance, the blue cut exists through the edge (A_2, A_3) has a cost of $C = |d' - d_A(3)|$. Note that, by definition, the edge (A_2, A_3) is left out of the total cost.

refining these candidates.

$$C_3 = \{d_{(x,y)}, d_{(x,y)} - .5, d_{(x,y)} + .5, d_{(x,y)} + .25, d_{(x,y)} - .25, d_{(x,y)} - .75, d_{(x,y)} + .75\} \quad (13)$$

The texture parameters are defined to break the smoothness constraint at image edges. We chose the following edge aware model:

$$\alpha_{p,q} = \frac{1}{1 + 30\|I(p) - I(q)\|} \quad (14)$$

The difficulty of the method is to find the right candidates. If the right candidate is missing, then the method cannot recover. The results indicate however that candidates are rarely missed.

5 Results

Results of the filtering process are displayed in Fig. 5 and Fig. 6. On Fig. 5 The left and right views are on top. On Fig. 6, the left column shows the left view image; the middle column the initial rough depth estimation; the refined map is on the right. Self-occlusions are displayed in black. Note that the refined map is visibly smoother and that the jitter artefacts have been completely removed. The quality of the results is however still essentially limited by the likelihood model, and oversmoothing of the depth map is sometimes noticeable across depth boundaries.

Comparison with full graph processing. On Fig. 7, we compare the candidate approach to using a full graph. On the right the results with using a full graph and on the left the results with the candidate approach described in this paper. The quality of both techniques is globally similar, although, on some places the candidate based approach displays some artefacts. These are due to a failure to pick up the correct disparity as a candidate. The candidate scheme has however a number of significant advantages: 1) it generates a subpixelic disparity map, 2) the computational effort is much lower.

On a 2.2GHz machine and using Kolmogorov's implementation of maximum-flow/min-cut, graph-cut takes around a 2 seconds to process a 300x300 image block with 8 disparity candidates. Overall, including the DP pass, our technique takes around 25 second to process the 400x300 sequences. The computational time is independent of the number of disparities. The memory load is around tens of MB and is also constant.

With the full graph mincut, the processing time takes around 20 to 30 seconds for 20 disparities but for 60 disparities, the processing time raises to 2-3 minutes and the memory usage to 1GB. Dealing with higher resolution pictures and wider disparity range becomes quickly intractable.

6 Conclusion

We have presented an algorithm that allows to deal with non-regular values in Ishikawa's and Roy's graph. The graph

structure can be used to resolve depth estimation MRF's energies for a sparse set of depth candidates. It offers thus an efficient framework to resolve multiple depth candidates per pixel at once in a reasonable amount of time. The drawback is that the smoothness penalty has to be convex and may oversmooth the depth map. We employed this technique to refine rough depth estimations obtained from using dynamic programming along scanlines. We showed that the method is very effective at improving the spatial smoothness of the rough estimates and can produce high quality continuous depth maps at a low computational cost.

A C++ implementation of the graph setup for sparse candidates is provided at www.sigmedia.tv/Publications/CVMP09/.

Acknowledgements

This work was funded by the FP7 European Project i3Dpost.

References

- [1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, Nov 2001.
- [2] Hiroshi Ishikawa. *Global optimization using embedded graphs*. PhD thesis, New York, NY, USA, 2000. Adviser-Geiger, Davi.
- [3] Sylvain Paris, François Sillion, and Long Quan. A surface reconstruction method using global graph cut optimization, February 2006.
- [4] Thomas Pock, Thomas Schoenemann, Gottfried Graber, Horst Bischof, and Daniel Cremers. A convex formulation of continuous multi-label problems. In *European Conference on Computer Vision (ECCV'08)*, 2008.
- [5] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.
- [6] Sébastien Roy. Stereo without epipolar lines: A maximum-ow formulation. *International Journal of Computer Vision*, 34(2/3):147–162, August 1999.
- [7] Sébastien Roy and Ingemar J. Cox. A maximum-ow formulation of the n-camera stereo correspondence problem. In *Proceedings of the International Conference on Computer Vision*, pages 492–499, January 1998.
- [8] O.J. Woodford, P.H.S. Torr, I.D. Reid, and A.W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

(a) Left View

(b) Right View

(c) Rough Depth Map

(d) Refined Depth Map

Figure 5: Example of Depth map filtering using the proposed candidate selection. On (a), (b), the left and right views. The original DP depth map approximation on (c). The results using candidates from the rough estimate on (d). Occlusions are marked in black. Pictures courtesy of The Foundry.

(a) Left View

(b) Rough Depth Map

(c) Refined Map

(d) Left View

(e) Rough Depth Map

(f) Refined Map

(g) Left View

(h) Rough Depth Map

(i) Refined Map

(j) Left View

(k) Rough Depth Map

(l) Refined Map

Figure 6: On the left, the left view. On the middle, the initial depth map approximation. On the right the results using candidates from the rough estimate. Occlusions are marked in black. Pictures courtesy of The Middlebury database and The Foundry.

Figure 7: *On the left, the disparity results on the Middlebury test images using the full graph of Roy and Ishikawa. On the right, the disparity results using the proposed candidate selection scheme. A few artefacts are visible, but subpixelic disparities can also be obtained.*