Fast Multi-labelling for Stereo Matching

Yuhang Zhang[1], Richard Hartley[1 2], and Lei Wang[1]

[1]The Australian National University
[2]NICTA
{yuhang.zhang,richard.hartley,lei.wang}@anu.edu.au

Abstract. We describe a new fast algorithm for multi-labelling problems. In general, a multi-labelling problem is NP-hard. Widely used algorithms like α-expansion can reach a suboptimal result in a time linear in the number of the labels. In this paper, we propose an algorithm which can obtain results of comparable quality polynomially faster. We use the Divide and Conquer paradigm to separate the complexities induced by the label set and the variable set, and deal with each of them respectively. Such a mechanism improves the solution speed without depleting the memory resource, hence it is particularly valuable for applications where the variable set and the label set are both huge. Another merit of the proposed method is that the trade-off between quality and time efficiency can be varied through using different parameters. The advantage of our method is validated by experiments.

1 Introduction

Solving multi-labelling problems by way of Markov random field (MRF) optimization has been a popular research topic in recent years due to its effectiveness. Successful algorithms like α-expansion [1, 2] and FastPD [3, 4] can already provide high quality solutions in polynomial time. However, most existing work involved images of relatively small size or with a limited number of labels. From the perspective of practical applications, for example large-scale 3D reconstruction based on high resolution images, both the number of variables and the number of labels involved in the optimization will be huge. Therefore, developing an even faster optimization method that is scalable to the size of the Markov random field attracts our attention.

One important attribute of the variables in a Markov random field is that their values (or labels) are generally smooth, whereas violent fluctuation only happens in sparse boundary areas. Based on this assumption, we divide each multi-labelling problem into two smaller multi-labelling problems, which handle the huge variable set and the huge label set respectively. In the first subproblem, we force some of the neighbour variables to share the same labels, so the number of random variables is reduced. In the second problem, we fine tune the result of the first subproblem, so that neighbouring pixels that were forced to share identical labels can now have different labels. During this fine tuning we exclude the possibility of violent variations, so the number of optional labels for each

variable is largely reduced. Through implementing these two steps recursively, the original problem can be solved in a time sublinear in the number of labels.

High speed and low memory cost is the major contribution of our method. However, rather than exclusively pursuing speed, a trade-off between efficiency and quality can be tuned by varying the parameters in our method. Experiments show that our method possesses strong advantages in efficiency over previous algorithms.

2 Previous Work

Many low level computer vision tasks can be modelled as multi-labelling problems. Examples include stereo matching, image denoising and image segmentation. A multi-labelling problem can be solved by energy minimization. In particular, Markov Random Fields have been popular as a formulation for multi-label energy minimization. The Hammersley-Clifford theorem makes the connection between the probabalistic viewpoint of MRFs and minimization of an abstractly defined cost-functions.

As optimizing a multi-label Markov random field is in general NP-hard [5], all usable algorithms need to sacrifice optimality for time efficiency [1, 4, 6–8] or generality. Among current approaches, α-expansion [1] is one of the most popular algorithms. The divide and conquer paradigm is utilized in α-expansion. Specifically, instead of tackling the intractable full problem directly, α-expansion assumes that the applicability of each label can be checked individually. Therefore, each n-label problem is divided into n 2-label subproblems, each of which determines whether a label α should be applied to variables currently having other labels. Each 2-label problem can be solved with a max-flow algorithm in polynomial time provided the edge costs satisfy a metric [1] or convexity [6] condition. To reach a stationary point, or local minimum, multiple outer iterations are required. One of the most popular max-flow algorithm, Push-relabel, has a time complexity of $O(M^3)$, where M is the number of vertices (variables) in the graph. Assuming the number of variables and the number of labels are independent, and that the number of necessary outer iterations is a constant, the time complexity of α-expansion is linear in the number of labels and the time complexity of its max-flow subroutine, namely $O(M^3N)$, where N is the number of labels.[1] However, in applications like stereo matching, the number of labels is usually proportional to the image resolution. Hence the processing time of α-expansion increases rapidly as the image size grows. A restriction on using α-expansion with max-flow is that the 2-label sub-problems must be submodular; this constrains its applicability to cost functions with metric or convex edge costs. However, there is no need to restrict the method for solving the 2-label problems to max-flow. Other algorithms such as roof-dual (QPBO) [10] or Lazy-Elimination [7] can be used instead. Another method that handles more generic

[1] Push-relabel is not the fastest max-flow algorithms. The fastest one so far was proposed in [9] and has a time complexity of $O(\min(M^{2/3}, E^{1/2})E\log(M^2/E+2)\log C)$, where C is the maximum capacity of the network and E is the number of edges.

energy functions is Alphabet Soup [11], which can be viewed as a generalized α-expansion.

FastPD [3] is another very successful multi-labelling algorithm. By relaxing the integrality constraint to a continuous positive constraint, it transforms the original discrete optimization problem into linear programming and then approaches the optimal solution through iteratively applying the primal-dual schema. FastPD can handle quite general types of energy functions and also obtains a convincing suboptimal results very rapidly. However, to pursue high processing speed, the avaliable implementation of FastPD easily exhausts the memory resource even dealing with a middle-sized stereo pair (e.g. 600×500 containing 64 labels). A more comprehensive review of relevant algorithms can be found in [12, 13].

Earlier work on enhancing the speed of MRF optimization can be found in [8], where the max-flow problem on a graph is efficiently solved using the known solution for a similar graph. Approaches to accelerating stereo matching were discussed in [14, 15], where higher speed (2.8 and 4 times speed-up respectively) is obtained through reducing the search range during matching. Particularly, an idea similar to our work was interpreted as search-range reduction through downscaling, tested and reported to fail in [15]. However, that paper gave only a sketchy description of their algorithm, with insufficient detail for us to distinguish the cause of failure. In contrast, our work shows that through proper construction and implementation, this method can handle MRF optimization with promising increase in a speed significantly faster than the existing algorithms.

3 Dividing the Original Problem

Optimizing a Markov random field containing M variables and N labels can be described as finding the optimal among N^M discrete states. For positive integers M and N both of which are larger than 1, and positive integers m and n which are smaller than M and N respectively:

$$\begin{aligned} & N^{M-m} - 1 \geq n^{M-m} \\ \Rightarrow\ & n^m(N^{M-m} - 1) \geq n^M \\ \Rightarrow\ & N^m(N^{M-m} - 1) > n^M \\ \Rightarrow\ & N^M - N^m > n^M \\ \Rightarrow\ & N^M > N^m + n^M \end{aligned}$$

Hence the complexity of the original problem can be reduced, if we can divide the original problem, i.e. split M and N into separate subproblems and tackle them respectively.

To construct the first subproblem, we need to reduce the number of random variables. On a 4-connected Markov grid as shown in Figure 1, a minimum cycle is composed of 4 vertices. Each vertex is directly related to 2 out of 3 of the other vertices on the cycle. Hence the values of all vertices on the same cycle

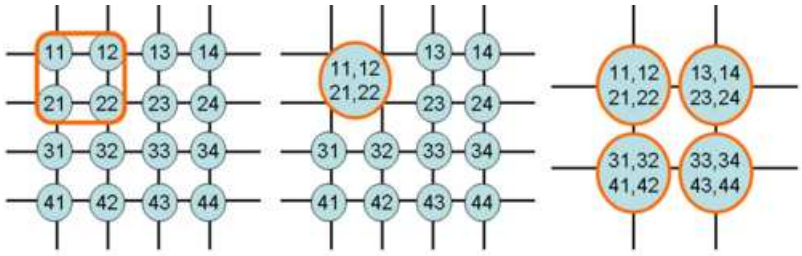

Fig. 1. *left: the original Markov grid; middle: four vertices on the same minimum cycle are merged, their exterior edges are inherited by the super-vertex; right: merging has been implemented with the rest vertices, some duplicate edges connecting the same super-vertices are dropped off.*

should generally be similar. Based on this prior, we favour vertices on the same minimum cycle to share the same value. Note that we cannot force vertices on all the minimum cycles to share the same value, otherwise all the vertices will have identical value. Whereas each vertex participates in 4 minimum cycles, we only force it to share the same value with neighbour vertices in 2 of the cycles. In this way, vertices on the selected minimum cycles are merged into one vertex, which inherits all the exterior edges of its component vertices. By merging all vertices on selected minimum cycles, the number of vertices in the Markov grid is reduced to a quarter of the original. At the same time, the number of edges is also reduced to a quarter of the original. Half of the edges in the original rigid are removed as interior edges, whereas a further quarter are eliminated as duplicate exterior edges. The consequent Markov grid after merging inherits the general structure of the original but loses the local details.

In the second subproblem, we retrieve the details lost due to merging vertices in the first subproblem. Since the general structure of the Markov grid has already been identified, only fine tuning within a small range is necessary for each vertex. That is why we can reduce the number of labels in the second subproblem. The fine tuning range depends on the nature of the problem and the requirement of the user. The proposed method favors problems satisfying the following criterion. Given the labels of surrounding vertices, the possible labels for a vertex can be narrowed down to a subset of all labels. In many practical problems, most of the vertices can be regarded as satisfying the above criterion. In such cases, the loss in optimality is limited. Obviously, the loss in optimality can be reduced through increasing the fine tuning range. The highest quality, as well as the worst time efficiency, is achieved by using all labels in the fine-tuning. In this case, however, the result obtained in the first subproblem is useless, and the problem is solved in its original form in the second subproblem. At the other end of the scale, the best time efficiency is obtained by using as few labels as possible in the fine-tuning step, i.e $\{-1, 0, +1\}$.

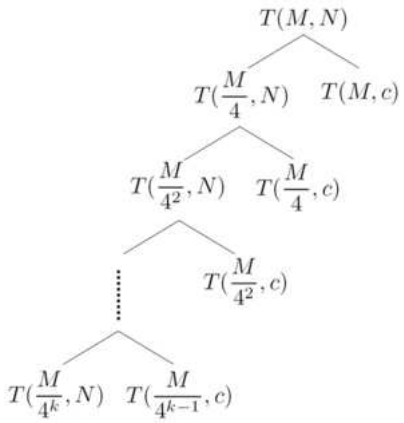

Fig. 2. *the original problem as well as its time complexity are divided recursively along the binary tree. The number of variables reduces as the tree branches.*

4 Recursion and Solving

Rather than dividing the original problem into two subproblems, our ambition really lies in recursively implementing the division. Theoretically, we can always further divide both the subproblems, as long as they are still large enough. However, in implementation, we generally use a small fine tuning range in the second subproblem for the purpose of time efficiency. Thus the second subproblem usually cannot be further divided. Therefore, the recursive division usually generates a binary tree which only branches on the left children as in Figure 2.

After dividing the original problem according to the binary tree in Figure 2, we solve the subproblems on the leaves and combine them together to generate the solution for the original problem. The time complexity of the proposed algorithm is the sum of the time complexity on all the leaves:

$$T(M,N) = T(\frac{M}{4^k},N) + \sum_{i=0}^{k-1} T(\frac{M}{4^i},c)\ . \tag{1}$$

Among all the leaves, only one of them is the left child of its parent and might have a large number of labels. However, since the number of variables shrinks rapidly as the tree branches, this single left leaf which is on the highest level of the tree contains extremely few variables. The time needed to solve it can be regarded as a small constant. All the other leaves are the right children of their parent, hence have a very small number of labels, which can be solved efficiently using any existing single-scale multi-labelling algorithm. In our work, we adopt α-expansion as the default option. Therefore, (1) can be computed as:

$$T(M,N) = \Theta(1) + \sum_{i=0}^{k-1} O((\frac{M}{4^i})^3 c)$$

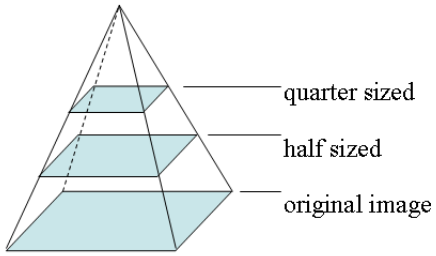

Fig. 3. *the image pyramid generated by the proposed method in stereo matching.*

$$\begin{aligned} &= \Theta(1) + \sum_{i=0}^{k-1} 64^{-i} O(M^3 c) \\ &< \Theta(1) + \frac{64}{63} O(M^3 c) \\ &= O(M^3 c) \end{aligned}$$

which shows the time complexity of the proposed method is independent of the number of labels N. As the fine tuning range c can be treated as a small constant independent of the size of the original problem, the time complexity of the proposed method is simply $O(M^3)$. Soon we will see that a dynamic c is even more powerful than a constant c.

Particularly in stereo matching, the above recursive division can be interpreted as depth estimation over a Gaussian-pyramid as shown in Figure 3. The bottom of the pyramid corresponds to the original image. Each division generates a higher level in the pyramid, where the numbers of variables and labels are reduced to a quarter and a half respectively. Optimization starts from the top level, where the number of variables and search range are both the smallest. The solution for the bottom level, namely the original image, is reached through hierarchical fine tuning.

5 Fine Tuning with α-expansion

As α-expansion calls max-flow algorithms as its subroutine, it is necessary for the binary term in the energy function to be submodular as shown in (2):

$$E(0,0) + E(1,1) \leq E(0,1) + E(1,0) \quad . \tag{2}$$

According to the original design of α-expansion [1], the above requirement is interpreted as

$$E(p,q) + E(\alpha,\alpha) \leq E(p,\alpha) + E(\alpha,q) \ . \tag{3}$$

In our problem the above interpretation is generalized to a more complex form:

$$E(\hat{p}+p,\hat{q}+q)+E(\hat{p}+\alpha,\hat{q}+\alpha)\leq E(\hat{p}+p,\hat{q}+\alpha)+E(\hat{p}+\alpha,\hat{q}+q)\ , \quad (4)$$

where $\hat{p}$ and $\hat{q}$ are the labels belonging to the two vertices before the fine tuning, namely the labels obtained in the first subproblem. The original α-expansion can be viewed as a special case in our problem, where $\hat{p}=\hat{q}=0$. To show what (3) really means, we discuss it in three complementary situations. To simplify the discussion, we assume the range of fine tuning is $\{-1,0,+1\}$, $\hat{p}-\hat{q}=d$, and that $E(p,q)=f(p-q)$.

- When $\alpha=-1,p=0,q=1$, equation (4) is converted to

$$f(d-1)+f(d)\leq f(d+1)+f(d-2)\ , \quad (5)$$

  requiring the energy function to be convex.
- When $p=-1,\alpha=0,q=1$, equation (4) is converted to

$$f(d-2)+f(d)\leq f(d-1)+f(d-1)\ , \quad (6)$$

  requiring the energy function to be concave.
- When $p=-1,q=0,\alpha=1$, equation (4) is converted to (5), requiring the energy function to be convex again.

According to the above discussion, neither concave nor convex functions can fulfill the requirement of submodularity in all situations. However, as has been shown by many previous papers, the Potts model [2] can usually do the job. As we can judge, it does fulfill the requirement of submodularity in all the three conditions when $d=0$. However, in our case, as d is no longer 0, the Potts model as well as many other energy functions longer guarantee the submodular requirement.

The solution lies in one particular function:

$$f(x)=|x|\ , \quad (7)$$

which is globally convex as well as sectionally concave. It is globally convex, hence (5) is naturally met. It is sectionally concave, when all points are sampled from the same side of the origin. Thus (6) can be met as well if we can ensure that the four terms lie on the same side of the origin. Obviously, if the two terms on the left lie on the same side of the origin, the two terms on the right will do so as well. The left two terms, to give their original form in (4), $(m+p)-(n+q)$ and $m-n$, are the label difference between two neighbour variables, before and after the fine tuning. Enforcing $((m+p)-(n+q))\,(m-n)\geq 0$ indicates that neighbouring variables should not invert their label orders in the fine tuning step. Although such a constraint confines the fine tuning moves, the optimality should not be affected much unless severe mistakes were made earlier, requiring such an inversion in the label orders.

An alternative implementation of α-expansion was proposed in [6], which requires the edge function to be convex in all the 3 situations. With this design,

more convex functions can be adopted in our problem. However in practical usage, if we expect the label jumps to happen intensively at narrow edges instead of loosely covering a wide area, concave functions are preferable to convex functions. Thus we use (7) as the smooth term in our method. This is a convex function by definition, but penalizes sharp edges the least among all the convex functions.

Note that α-expansion is not the only option for single-scale MRF optimization. We stick to it in this work because it is widely known. People familiar with other single-scale MRF optimization methods like [3] and [7] can adapt our methods into their work very easily without necessarily restricting to the cost function (7).

6 Experiments

To examine the usefulness of the proposed method, we use it to estimate the disparity between high resolution stereo image pairs. The machine is an average PC, equipped with 2 GB RAM and 2.39 GHz Dual Core CPU. The images are downloaded from the Middlebury stereo dataset [16, 17]. All the images are over 1000×1000 in size. The disparity range in each image pair is larger than 100 pixels, requiring more than 100 labels. We will use α-expansion as the subroutine in our method, and compare its performance against the original single-scale α-expansion. Both methods optimize the energy function given by (8), (9), (10), where $I_l(i)$ is the normalized grey value of pixel i in the left image, $I_r(i')$ is the normalized grey value of its corresponding pixel in the right image, and $L(i)$ and $L(j)$ are the labels of pixel i and j respectively. Certainly more sophisticated energy functions can be designed and used instead, which is an important research topic by itself, but not our major concern in this work.

$$E = \sum_i U_i + \frac{1}{100} \sum_{ij} B_{ij} \tag{8}$$

$$U_i = |I_l(i) - I_r(i')| \tag{9}$$

$$B_{ij} = |L(i) - L(j)| \tag{10}$$

We have also implemented our method with FastPD as the subroutine. However, as the single-scale FastPD cannot be executed to solve a problem of this size due to the memory limitation on a normal PC, we cannot conduct detailed comparison with it, but only show our output.

Table 1 shows the processing time, peak memory usage and result quality for the different methods. All values are the average over the 4 image pairs we used in the experiments. We refer to the original single-scale α-expansion as 1-scale in the table. 1-scale* is equivalent to 1-scale except that it is optimized for speed during programming and hence demands a much larger memory space than 1-scale. The other rows in the table correspond to our method implemented over different numbers of hierarchies. The tuning range in the second subproblems is

$\{-1, 0, +1\}$ in all cases. Figure 4 shows the left views of the four stereo pairs, the ground truth and the results produced by different methods. The content of the image are chosen to be different and representative. For example, the image in the second column contains many slender objects, which cause frequent and sharp discontinuities in depth value, whereas the image in the last column contains generally smooth surfaces on which depth value varies only mildly.

Method	**Time**	**Peak Memory**	**Extra Energy**(%)
1-scale	34m03s	270MB	0
1-scale*	17m01s	2.3GB	0
2-scale	2m21s	540MB	4.33
3-scale	37s	540MB	7.63
4-scale	27s	540MB	11.63
5-scale	22s	540MB	18.50

Table 1. *Performance comparison: 1-scale* follows the original α-expansion algorithm, except that it is optimized for speed during programming.*

6.1 Time and Memory Efficiency

The proposed method is absolutely faster than single-scale α-expansion, no matter whether it is optimized for speed or not. The optimized α-expansion uses arrays to store precomputed unary terms. Consequently, it requires 10 times the memory of the original α-expansion. Nevertheless, it only becomes 2 times faster. Moreover, when its memory occupancy exceeds the physical memory limit, virtual memory swapping makes it even slower. That is also why FastPD cannot be used here. On the other hand, the memory requirement of our method is only moderately larger than that of the original α-expansion, but the boost in speed is significant. With FastPD as the subroutine, the peak memory occupancy of our method is 1.4GB, and the algorithm terminates within 15 seconds over 3 levels of hierarchy.

Significant decrease in processing time can be observed between 1-scale and 2-scale, as well as between 2-scale and 3-scale optimization, but as we further increase the number of hierarchies, no further increase is observed. That is because, after merging the vertices once, the size of the first subproblem is still quite huge, whereas after merging twice or three times, the size of the first subproblem is already small enough that further reducing its size will not save much time.

After cutting the problem into small enough pieces, the peak memory occupancy, as well as the processing time of our method is determined by the final fine tuning on the original scale. That is why the peak memory occupancy of our method remains the same irrespective of the number of hierarchy levels. It needs

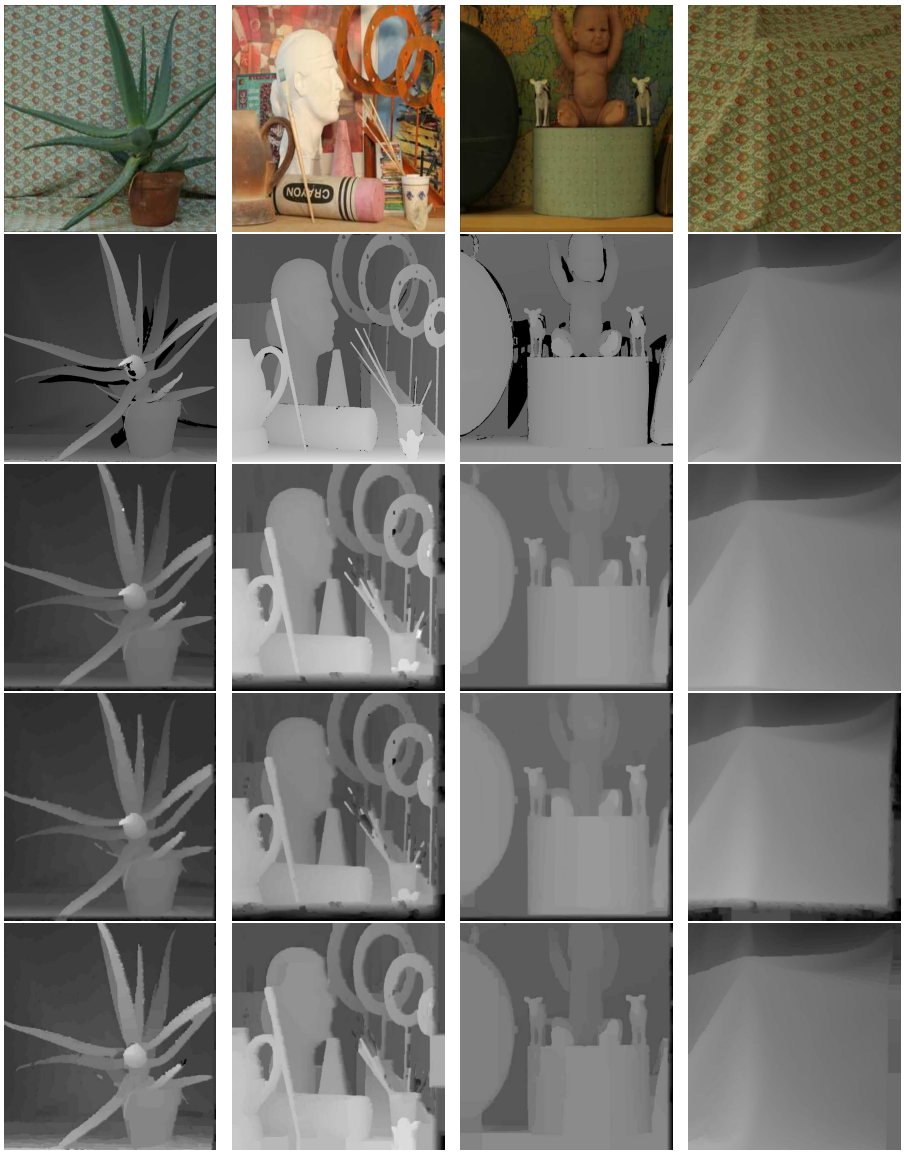

Fig. 4. *from the top to the bottom: left views of the stereo pairs, ground truth, result produced by the original α-expansion, the proposed method over 3 levels of hierarchy, using α-expansion and FastPD as subroutines respectively.*

more memory than the original α-expansion because we have pre-computed the unary term. However, since the number of labels in the fine tuning is small, the additionally required memory space does not become a problem for a normal PC.

That also suggests that on images of the same size but with increasing number of labels, whereas the running time of α-expansion or the memory occupancy of FastPD will increase accordingly, the cost of our method remains almost the same, because the number of labels needed in the final fine tuning is not changed.

The time and memory efficiency of single-scale α-expansion and FastPD may vary due to different implementations, however, that will not affect the comparison here, because the proposed method calls them as the subroutine.

6.2 Quality

As shown in the second column in Figure 4, our method does not handle slender objects particularly well. Slender objects correspond to a sequence of discontinuities in depth values. This result is not surprising, as our assumption is that neighbouring pixels have similar labels (depth values), and hence can be merged. Such an assumption does not apply to slender objects, where adjacent pixels may possess completely different labels. Merging in these areas leads to severely wrong labelling which cannot be corrected by small range fine tuning. That is why part of the stick is absorbed by the background, and the other part becomes thicker through absorbing background pixels. However, as slender objects are naturally difficult for graph cuts, even the original α-expansion does not perform outstandingly well on them.

Despite the above defects, the proposed method performs quite well with generally smooth surfaces like the cloth image in the last column, and sparse edges like the aloe image on the first column. The extra energy in the MRF due to hierarchical optimization, as shown in Table 1, is minor, as long as we do not use too many hierarchies.

Figure 5 shows how the final result is reached through sequential fine tuning from the coarse estimation. In particular, only the final three levels of recursion are shown here. The total number of recursion levels is 5. The images in the first row show the improvement as the disparity estimate becomes more and more accurate with increasing resolution. The images in the second row reflect the fine tuning operation on different pixels.

6.3 Trade-off between Efficiency and Quality

Table 1 shows that the trade-off between efficiency and quality can be tuned by changing the number of hierarchy levels of optimization. Another parameter affecting the trade-off is the range of fine tuning. As a rule of thumb, using more labels during fine tuning over all hierarchies will significantly increase the processing time. Consequently, we only increase the range of fine tuning in hierarchies other than the final one. Recall that the processing time of our method is mainly determined by the last fine tuning at the original scale. As long as the problem size of this step remains the same, the time efficiency of the whole algorithm will not be significantly changed.

Table 2 shows how time efficiency and result quality vary as we use different numbers of levels of hierarchy and different tuning ranges. As one would expect,

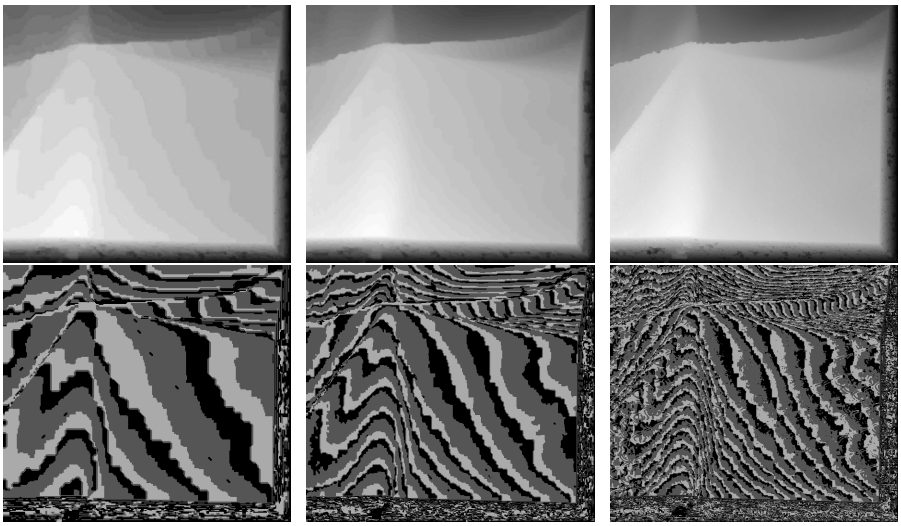

Fig. 5. *Refinement and convergence of depth values at different levels of resolution. The first row shows depth values. The second row shows which pixels change their labels as resolution is increased during the sequential fine-tuning operation. The pixels colored black are to be decreased in disparity as resolution increases; the pixels colored white are to be increased in disparity, and the pixels colored grey will keep their current disparity.*

with the same number of levels, the wider the tuning range is, the better the quality will be, and the slower the algorithm will be. However, the opposite results can be found when the tuning range is increased to 7, i.e. $\{-3, -2, -1, 0, +1, +2, +3\}$. Not only the energy in the MRF but also the processing time is reduced. Again, note that the processing time mainly depends on the last fine tuning in the original scale, which is a single-scale α-expansion. The processing time of a single-scale α-expansion algorithm depends not only on the size of the MRF but also on the initial state. The closer the initial state is to the optimal state, the fewer iterations the α-expansion algorithm needs to converge, hence the faster it terminates. Although using a wider tuning range takes more time on the earlier hierarchies, it also generates better initial estimation for the final hierarchy, which is the payoff for previous loss. Figure 6 visually compares the difference in quality due to different combinations of parameters.

7 Conclusion

The proposed method provides a mechanism for separating the complexity induced by the variable set and the label set. This mechanism is able to obtain satisfying optimization results in time much shorter than that of the other existing algorithms. This speed opens an opportunity for large scale MRF optimization. The tunable parameters leads to the versatility of our method, which can

Levels	Tuning Range	Time	Extra Energy(%)
3	3	37s	7.63
	5	40s	6.00
4	3	27s	11.63
	5	37s	8.75
	7	35s	7.53
5	3	22s	18.50
	5	37s	10.62
	7	32s	7.98

Table 2. *Efficiency and quality with different combinations of parameters. As expected the quality of the solution is improved by using a larger fine-tuning range. This is done at all levels of hierarchy, except at the finest resolution level. This improvement in quality is also at times accompanied by a decrease in run time. The table also shows that increasing the number of levels can be counter-productive.*

Fig. 6. *Result produced with different parameters using α-expansion as the subroutine. From left to right: 4 hierarchies, 3 tuning labels; 3 hierarchies, 3 labels; 3 hierarchies, 5 labels.*

be applied to different applications through proper parameter selection. For future work, we see the possibility of combining our method with Dynamic Graph Cuts [8], where segmentation in previous frames can be used to guide the formation and fine-tuning in the hierarchy. In another approach, to avoid improperly merging vertices of completely different values, mechanisms like backtracking might be adopted into the proposed method. With these improvements, fast algorithms producing results of even better quality are to be expected.

Acknowledgement

We acknowledge the support of NICTA, which is funded by the Australian Government in part through the Australian Research Council.

References

1. Zabih, R., Veksler, O., Boykov, Y.: Fast approximate energy minimization via graph cuts. In: ICCV99. (1999) 377–384
2. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23** (2001) 1222–1239
3. Komodakis, N., Tziritas, G., Paragios, N.: Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies. Comput. Vis. Image Underst. **112** (2008) 14–29
4. Komodakis, N., Tziritas, G.: Approximate labeling via graph cuts based on linear programming. PAMI **29** (2007) 1436–1453
5. Boros, E., Hammer, P.L.: Pseudo-boolean optimization. Discrete Applied Mathematics **123** (2002) 155–225
6. Carr, P., Hartley, R.: Solving multilabel graph cut problems using multilabel swap. In: DICTA, Melbourne, Australia (2009)
7. Carr, P., Hartley, R.: Minimizing energy functions on 4-connected lattices using elimination. In: ICCV, Kyoto, Japan (2009)
8. Kohli, P., Torr, P.: Dynamic graph cuts for efficient inference in markov random fields. PAMI **29** (2007) 2079–2088
9. Goldberg, A.V., Rao, S.: Beyond the flow decomposition barrier. J. ACM **45** (1998) 783–797
10. Rother, C., Kolmogorov, V., Lempitsky, V., Szummer, M.: Optimizing binary mrfs via extended roof duality. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2007)
11. Gould, S., Amat, F., Koller, D.: Alphabet SOUP: A framework for approximate energy minimization. CVPR (2009)
12. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. IEEE Trans. Pattern Anal. Mach. Intell. **30** (2008) 1068–1080
13. Scharstein, D., Szeliski, R., Zabih, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. SMBV '01: Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01) (2001) 131
14. Wang, L., Jin, H., Yang, R.: Search space reduction for mrf stereo. ECCV '08: Proceedings of the 10th European Conference on Computer Vision (2008) 576–588
15. Veksler, O.: Reducing search space for stereo correspondence with graph cuts. BMVC06 (2006) II:709
16. Hirschmuller, H., Scharstein, D.: Evaluation of cost functions for stereo matching. CVPR07 (2007)
17. Scharstein, D., Pal, C.: Learning conditional random fields for stereo. CVPR07 (2007)