DENSE STEREO MATCHING USING MACHINE LEARNING

Nattamon Thavornpitak Pallabi Ghosh Ayesha Khwaja

Introduction

Many researches in computer vision have been focused on developing algorithms to accurately determine depth maps. In stereo vision, a pair of cameras at two different locations capture a left and a right image which are slightly different images of the same scene. Due to different locations of the cameras, a 3D point will show up at different pixels in each image. To perform stereo matching is to map a pixel in the left image to the pixel in the right image that is corresponding to the same 3D point. The difference of locations of matched left and right pixel is called disparity. Disparity is inversely proportional to depth. An image can have only a finite number of objects and hence a finite number of disparity values. Many algorithms such as SIFT have been developed to perform sparse matching, which produce matches for only a few keypoint pixels. However in some applications such as robots' navigation, it is better to have disparity value and depth for every single pixel in an image. In this project we explore an application of machine learning in performing dense matching which will produce matches for all pixels in the images.

Background

Sparse Matching Technique

Sparse matching algorithms produce disparity values for keypoints. There are many sparse matching techniques such as matching by using Scale Invariant Feature Transformation (SIFT). In this project, we use the algorithm developed by Geiger et al. to produce disparity values for keypoints, which will be used as our training data[1]. Geiger's algorithm first preprocesses images by filtering input images with 5x5 blobs and corner mask and then applying non-maximum- and non-minimum-suppression[2]. Speed Up Robust Features (SURF) descriptors of each pixels are computed and the descriptors are passed to a Sobel filter. For a given keypoint pixel in the left image, sum of absolute differences (SAD) between its filter response and the filter response of every pixel in the right image are computed and the pixel of the right image that produces the smallest SAD is picked as the match.

DAISY

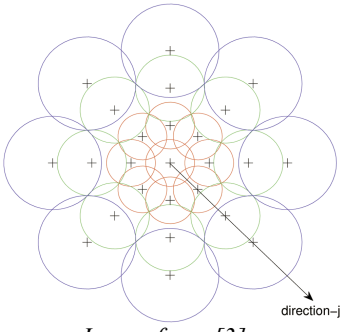

Image from [3]

Daisy is a computationally efficient dense descriptor. In this technique initially the gradient at each pixel location in different directions is calculated. This can be represented by $G_o(u;v) = \max((\delta I/\delta o),0)$, where o is the direction of the gradient, and I is the intensity. Next we convolve this with Gaussian kernels of varying standard deviation ie Σ_1 in one layer (a layer is a group of circles with center equidistant from the pixel in consideration) around each pixel and Σ_2 in another layer and so on. We use a 3 layered version of Daisy as shown in the figure (same as used by the developers of Daisy: E.Tola, V.Leptit and P.Fua in their paper[3][4]). In the figure, the Gaussian smoothing is proportional to the circle radius. Also this entire figure computes the daisy parameters for the central pixel. The overlapping circles result in smooth results.

Overview of Our Approach

We explore applications of learning algorithms including multi-class support vector machine(SVM) and K-mean clustering algorithm. We use Geiger's matching algorithm to generate disparity values for keypoint pixels. These pixels and their corresponding feature vectors are used to train our algorithm. The disparity values for rest of the pixels in the image are calculated using our trained model. Our approach is justified since for a given image, the number of disparity values is finite, which would most likely be covered in one of the training pixels. We initially use daisy parameters as our feature vector. Later on we add the normalized pixel location(x, y coordinate) to the daisy features. In addition, to generate some training examples in smooth regions, we performed pixel padding according to a heuristic that nearby pixels are likely to have the same disparity value. Finally, we try increasing the training size by finding the disparity values of edge pixels using correlation based matching.

Multi-class Support Vector Machine

Our approach is to use Geiger's matching algorithm to generate disparity values for keypoint pixels. These disparity values are used as class labels (target variable). Each keypoint pixel in the left image of a given pair of images is a training example. For a training example i in the training set of size $\tilde{m}$, we have a feature vector $\tilde{x}^{(i)}$ and a class label $\tilde{c}^{(i)}$, the pixel's disparity value calculated from Geiger's algorithm. We use these keypoints pixels and their disparity values to train our multi-class SVM model. Our algorithm extracts features from each of the remaining pixels in the left image, and uses the trained SVM model to predict the disparity value. In this report, non-keypoint pixels in the left image are referred to as pixels in a test set or test pixels. In this project, two sets of features are used and compared. While the first set of feature consists of 200 Daisy descriptors of a pixel, the second set uses both Daisy descriptors and the normalized x and y coordinates denoting the location of the pixel. Let n be the length of each feature vector then $\tilde{x}^{(i)}$ is a n x 1 vector. Therefore, a training set consists of a $\tilde{m}$ x 1 label vector $\tilde{C}$ and a $\tilde{m}$ x n feature matrix $\tilde{X}$.

$$\tilde{C} = \begin{bmatrix} \tilde{c}^{(1)} & \tilde{c}^{(2)} & \dots & \tilde{c}^{(i)} & \dots & \tilde{c}^{(\tilde{m})} \end{bmatrix}^T$$

$$\tilde{X} = \begin{bmatrix} \tilde{x}^{(1)^T} \\ \tilde{x}^{(2)^T} \\ \dots \\ \tilde{x}^{(i)^T} \\ \dots \\ \tilde{x}^{(\tilde{m})^T} \end{bmatrix}$$

The training set is input to multiclass-support vector machine available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

The test set consists of m non-keypoint pixels of the left image. To produce a dense map, we calculate feature vector $x^{(i)}$ for all m test pixels and use the trained model to predict the disparity value, $c^{(i)}$ of that pixel.

Modified K-mean Clustering Algorithm

In this approach, we modify K-mean clustering and apply it to our data. K-mean clustering algorithm is a learning algorithm for unlabeled data; however, in our scenario, pixels in the training set are labeled with disparity values, but test pixels are not. For each training example in the training set of size $\tilde{m}$, we have a feature vector $\tilde{x}^{(i)}$ and a class label $\tilde{c}^{(i)}$, which is ~~a~~ the disparity value calculated from Geiger's algorithm. We modify the algorithm as follows

Initialization : Number of clusters, k is the number of distinct disparity values in the training set. The initial centroid for the j[th] cluster is the average of features vectors of all pixels in the training set that are labeled as the j[th] class.

For each class j, μ_j, the initial centroid of the j[th] class is

$$\mu_j = \frac{\sum_{i=1}^{\tilde{m}} 1\{\tilde{c}^{(i)} = j\}\tilde{x}^{(i)}}{\sum_{i=1}^{\tilde{m}} 1\{\tilde{c}^{(i)} = j\}}$$

Iterations :

Step 1: For each pixel in the test set, compute Euclidean distance between $x^{(i)}$, feature vector of the pixel and all cluster centroids μ_j. Assign the pixel to cluster $c^{(i)}$ that has the smallest Euclidean distance.

For all pixels in the test set, determine $c^{(i)}$

$$c^{(i)} = \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

Step 2 : Update the cluster centroids by averaging the feature vectors of all pixels in the cluster(include both pixels in training and the test set).

For all clusters,

$$\mu_j = \frac{\sum_{i=1}^{\tilde{m}} 1\{\tilde{c}^{(i)} = j\}\tilde{x}^{(i)} + \sum_{i=1}^{m} 1\{c^{(i)} = j\}x^{(i)}}{\sum_{i=1}^{\tilde{m}} 1\{\tilde{c}^{(i)} = j\} + \sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

$1\{A\}$ is an indicator function which is equal to 1 if the statement A is true and is equal to zero otherwise. Both steps are repeated until the locations of cluster centroids converge, that is when the norm of the change of cluster centroids of two successive iterations is smaller than some tolerance value *tol*. Note that unlike K-mean clustering algorithm, the labels for pixels in the training set do not change.

Pixel Padding

Motivation

It is likely that all pixels that belong to the same object have the same disparity value since disparity values are proportional to depth and all pixels in an object are likely to have approximately the same depth. In addition, it is very likely that the size of an object is larger than one pixel. Therefore, if a given pixel has disparity value c, it is likely that the pixels next to it has the same disparity value since the next pixels probably belong to the same object. In addition, consider a scenario that a pixel A has a disparity value c and another pixel B, which is a few pixels apart from A, has the same disparity value. In this case, it is probable that the pixels in between A and B have disparity value c.

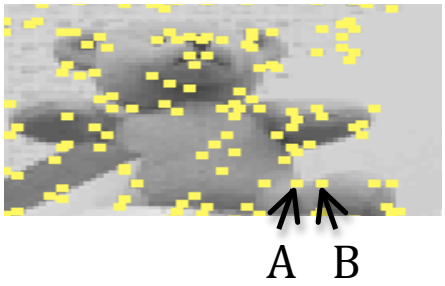

Fig.1

Pixel Padding

Sparse Matching algorithm produces matches for only a few keypoint pixels. These keypoints are mostly edges and corners. Yellow dots in Fig. 1 are keypoint pixels detected by Geiger's algorithm. Keypoint pixel A is said to be "adjacent" to keypoint pixel B if pixel B is the closest keypoint pixel(there is no keypoint pixel in between them on the same horizontal line). Note that pixel A and B are not next to each other but they are the closest keypoint pixels. The trick is that if any two adjacent keypoints have the same disparity value, it is likely that pixels that lie between the two keypoint pixels are from the same object and they should have disparity values c. We can assign all in-between pixels to have the same disparity value c; however, if we do that we will be relying on the assumption that all pixels between A and B are in the same object. If A and B are very far apart then the assumption may not be valid. Therefore, every time we have adjacent pixels that have the same disparity values c, we first check the distance between the two pixels. For example, if the distance is smaller than Dmax =15 pixels, we will assign a label c to in-between pixels that are not further than lmax=5 from either A or B. Fig 2 demonstrates this trick. Keypoint pixels are shown in yellow and non-keypoint pixels are shown in white. For a pair for adjacent keypoint pixels A and B that have the same disparity value c and are fewer than Dmax pixels apart, we assign disparity value c to pixels C, D, E, F and G which are in between A and B and are within lmax= 5 pixels from A. Similarly, we also assign disparity value c to pixel J,K,L,M and N since they are in between A and B and they are with lmax=5 pixels from B. Note that pixel H and I are not assigned the disparity value c because they are further than lmax pixels from both A and B.

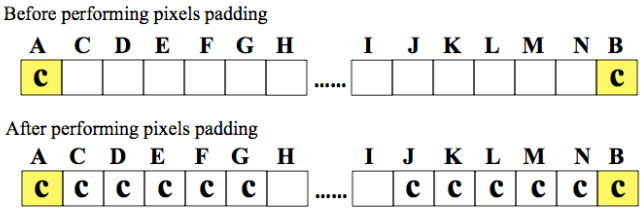

Fig.2

By applying this trick, we can generate some examples of pixels in smooth regions.

Correlation Based Matching

In order to generate a relatively larger training size, we use correlation based matching to generate more training examples. We use block by block correlation based matching on the edge pixels. We use the canny edge detector to get the edges

in the stereo pair. For each of the edge pixels, we search for a small patch of area around that pixel in the left image and try to match it to a similar patch around different pixels in the right image along the same horizontal line. The sum of norm-square of the difference in the intensities of each pixel in the left patch and its corresponding pixel in the right patch (i.e., the correlation between the two patches) gives us a measure of correspondence. We calculated the correspondence between the patch in the left image to different patches in the right image. The pivot pixel of the patch in the right image with the maximum correspondence is our closest match to the given pixel in the left image. Once this is done, difference between these pixel locations gives us the sparse disparity map. Extra training examples generated by correlation based matching are appended to the training set.

Results

The following images are results obtained by using SVM classification and modified K-mean clustering algorithm along with other techniques presented above. Please note that in an image, the same shade represents the same disparity value (white means the object is the closest and black, the object is farthest). Parameters (as in described in our approach section) used to generate these images are: Image size=350 x 292, tol=1, Dmax=15, lmax=5

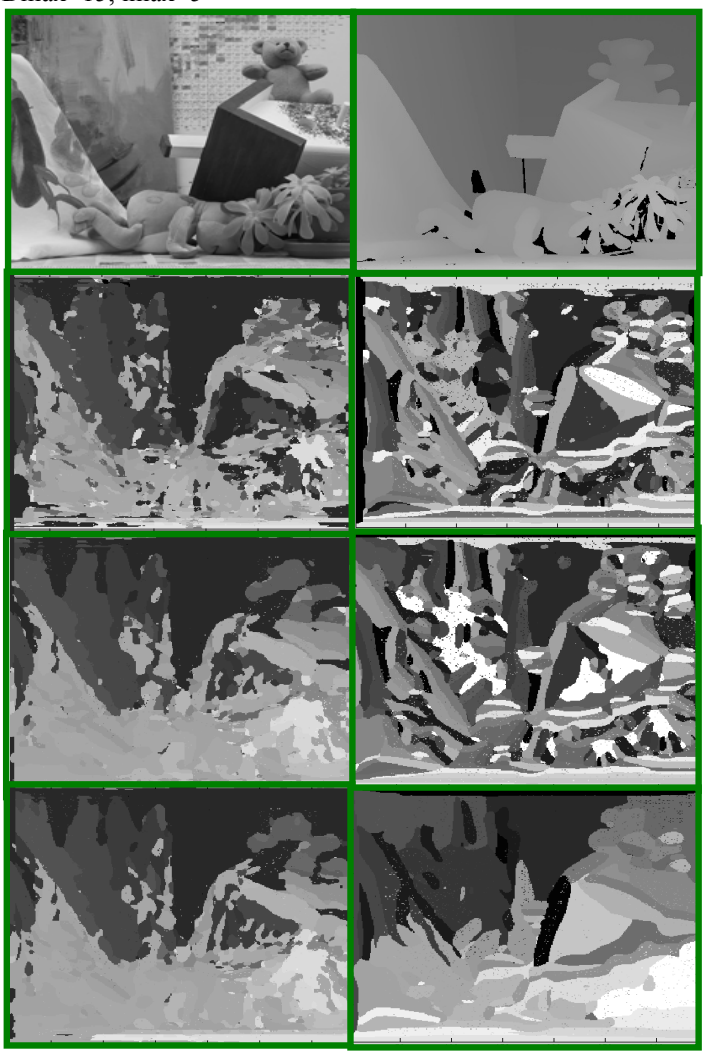

Fig. 3 Demonstration of the effect of using pixel location and pixel padding. The first row is the image and the groundtruth. The left column is the result obtained by SVM and the right column is the result obtained by Kmean algorithm. The second row is the result generated when neither pixel location nor pixel padding is used. The third row is the result when pixel padding is used but not pixel location. The last row is the result when pixel location is used but pixel padding is not used.

Fig. 4 Demonstration of the effect of using correlation based matching to generate more training examples. The first row is the image and groundtruth. The left column is the result from SVM classification and the right column is the result from K-mean clustering algorithm. The middle row is the result obtained when pixel location and pixel padding is used. The bottom row is the result obtained when pixel location, pixel padding and correlation based matching is used.

Fig. 5-8 are results obtained using SVM and k-mean clustering algorithm. The top row is the image and groundtruth and the bottom row is disparity value obtained from SVM(left) and K-mean clustering algorithm(right). For SVM, we used pixel location, pixel padding and correlation based method. For K-mean algorithm, we used pixel locations feature vectors (in addition to Daisy) but not pixel padding and correlation techniques.

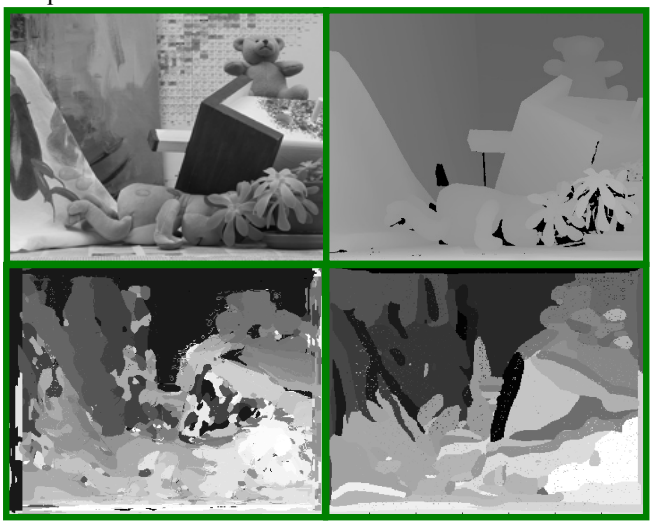

Fig. 5

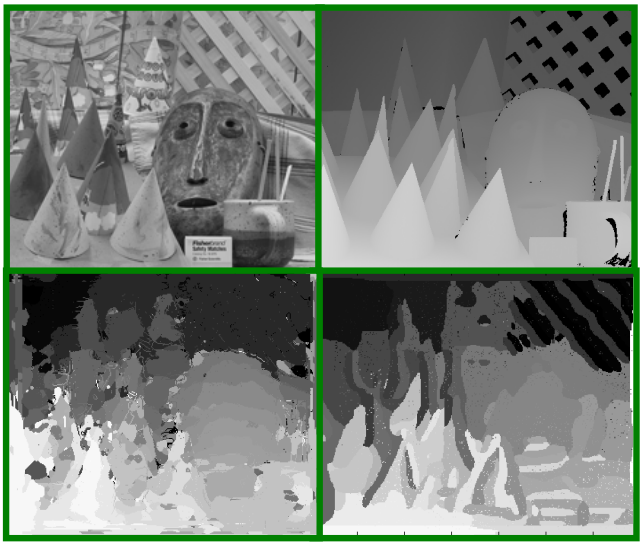

Fig. 6

Fig. 7

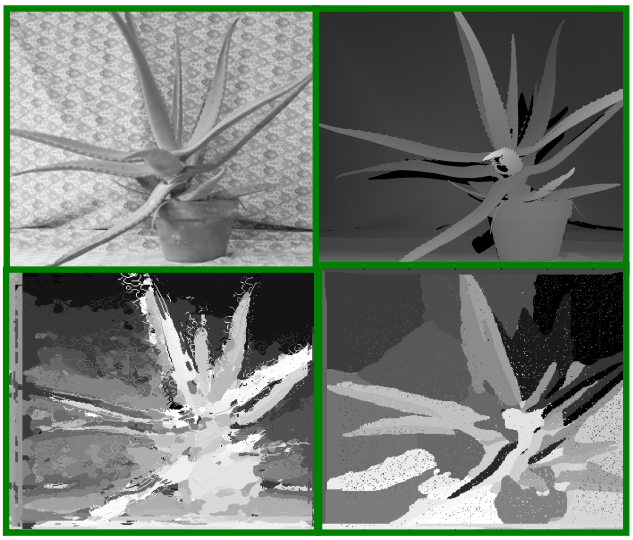

Fig. 8

Discussion

The first problem that arises in both SVM and K-mean clustering algorithms is that we fail to accurately determine disparity values of smooth regions. The reason is that all sparse matching features look for detectable positions in the images that can be exactly matched for stereo correspondence. So they detect edges and corners. Therefore, we do not have enough training examples in these smooth regions. To address the problem, we use the pixel padding trick which helps generate some training examples in smooth regions. For SVM, the trick improves the quality of dense map in smooth region, but it does not get rid of the problem(see Fig. 3, the third row). However, K-mean clustering algorithm does not incorporate the extra information provided by these pixels well. Therefore, for K-mean algorithm, the trick does not improve the result. To make these statements, we have applied pixels padding to several images and compared the results. (However, due to page limit, those results are not shown here).

In addition, we use pixel location as well Daisy descriptors as features. In the case, when our algorithm needs extra information apart from Daisy descriptors in order to accurately find the best match, adding pixel locations helps improve the results. As shown in Fig 3, for both SVM and K-mean algorithms, pixel's locations do mitigate the problem but does not solve it. Therefore, in future research we would suggest adding other features that are correlated with disparity values and studying the effect of those extra features more carefully.

Another modification we have tried is adding training pixels obtained by correlation based matching technique. For SVM, we have obtained a better disparity map (see Fig. 4). This is because we have a larger training set. Again, K-mean clustering algorithm does not incorporate the extra information provided by these pixels well, so this modification does not improve the results.

For multi-class SVM algorithm, another issue is the smoothness of a resulting disparity map and noise. Depth and disparity values of all pixels that belong to the same objects should either be the same or change gradually; therefore, we expect the disparity map to have some sharp edges corresponding to boundary of each object and some smooth regions inside an object. However, we observe that disparity maps produced by our SVM algorithm are quite noisy and edges are not sharp.

Compared to disparity maps returned by SVM, maps produced by modified K-mean clustering algorithm have sharper edges and smoother regions inside each object. However, there are three major issues. First, K-mean clustering algorithm does not accurately predict disparity values when images have a lot of details such in Fig. 7. Consider Fig. 7 the background is a bookshelf and a board which both have the same disparity value but quite different Daisy descriptors due to different color, intensity etc. Since the K-mean clustering algorithm classifies a pixel based on Euclidean distance between a feature vector (which includes Daisy parameters) of a given pixel and cluster centroids, it is more prone to this kind of error than the SVM. Fig.7 shows that in this case, the SVM algorithm does a lot better. The second issue for K-mean clustering algorithm is how to weigh each attribute. K-mean clustering algorithm relies on the calculation of Euclidean distance, which depends on how much weight is given for each entry of a feature vector. For example, in a feature vector, we can scale all Daisy parameters by a factor of 2 and that will give more weight to the Daisy parameters and change Euclidean distances. In this project, for a feature vector we use the ratio x/W and y/H coordinate which indicates location of a pixel. x and y denote horizontal and

vertical location of a pixel and W and H denote the width and the height of an image. Daisy parameters are scaled such that the average norm of DAISY vector of all examples in the training set is 1. The same scaling is applied to pixels in the test set. The issue of scaling each attribute requires a more careful study, which we plan to do later. The last issue is that when we perform pixel padding and adding the results of correlation based matching, K-mean algorithm does not incorporate this extra information well. This is due to simplicity of K-mean clustering which makes decision based on Euclidean distances.

One important observation is that the performance of our algorithms depends on the quality of the training set. Similar to all machine learning algorithms, if training examples are noisy and we do not have enough training examples, the learning algorithm cannot perform well. However, this issue should not be a problem since nowadays there exist various algorithms that can efficiently and accurately determine sparse matching. In addition, due to advances in computer vision technology, various algorithms such as Geiger's algorithm can produce sparse matches with larger number of keypoints.

Suggested Future Works

In this section we suggest some modifications that will potentially improve the performance of the stereo matching algorithms. We plan to implement these algorithms later. First, to solve the smooth region problem, some computer vision techniques may be used to detect smooth regions and these regions can be handled separately. Another way is to use some computer vision techniques to calculate an indicator, which indicates that a given pixel belongs to one of the smooth regions, and use this indicator as one of the attributes in the feature vector. This indicator should enrich the information in feature vectors that are used to predict disparity values.

The second modification is to modify the algorithms to exploit information from the right image. Sparse matching algorithms take pixels values for both left and right image to predict disparity value of all keypoints. Our algorithms use disparity values of these keypoints and features extracted from keypoint pixels in the left image as a training set. Information of pixels in the right image is not used in our algorithms. Therefore, the algorithm should be modified to take into account information from the right image. One possible way to do this is to classify a pair of left and right pixel as "a true match" or "a false match". Given a pixel in the left image, instead of using its feature to predict a disparity value, we can pair this left pixel to every right pixel. For each pair, we extract feature from both left and right pixel and from this feature we classify whether or not the pair is a true match, which means they are corresponding to the same 3D point. The feature vector of a pair of pixels can be either Daisy descriptors of both left and right image together with pixel location or norm of difference of Daisy parameters. By formulating the problem this way we can exploit the information from right image pixels. However, in this case we will have to search for other learning algorithms that are suitable for the new problem. For example, we should find an algorithm that picks the match that minimizes the difference between Daisy parameters of left and right.

Conclusion

We have presented various ways in which machine learning can be applied to perform dense stereo matching. The error of the disparity results obtained by these methods were calculated as the percentage of pixels differing from ground disparity values by more than 15, after leaving out some border pixels and occluded regions. The error percentage ranged from 20-40% for SVM and 40-60% for K-means. The main problem of our algorithm is smooth regions. Although our algorithms still have some issues, we have shown that machine learning is a potential way of performing dense stereo matching

Acknowledgement

We would like to thank Tao Wang for his advices and data.

References

[1] A. Geiger, J. Ziegler and C. Stiller, "StereoScan: Dense 3d Reconstruction in Real-time," in Intelligent Vehicles Symposium (IV), 2011

[2] A. Neubeck and L. V. Gool, "Efficient non-maximum suppression," in ICPR 2006, August 2006, in press.

[3] E. Tola, V. Lepetit, and P.Fua, "Daisy: An efficient dense descriptor applied to wide-
baseline stereo," in IEEE Transactions on Pattern Analysis and Machine Intelligence,
vol. 32, pp. 815-830, May 2010.

[4] Program is developed based on DAISY code - http://cvlab.ep.ch/software, 2008.

[5] D. G. Lowe, "Distinctive image features from scale invariant keypoints," in International Journal Of Computer Vision, vol. 60, pp. 91-110, Nov 2004.

[6] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two frame stereo correspondence algorithms," in International Journal Of Computer Vision, vol. 47, pp. 7-
42, Apr-June 2002.

[7] D. Scharstein and R. Szeliski, "High accuracy stereo depth maps using structured light," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), vol. 1, pp. 195-202, June 2003.

[8] Images used in this project have been downloaded from the website http://vision.middlebury.edu/stereo/

[9] Svm code: http://www.csie.ntu.edu.tw/~cjlin/liblinear/

[10]Sparse matching code:
http://www.cvlibs.net/software/libviso/Sparse matching