SEGMENTATION AND RECOGNITION IN DRIVING VIDEOS

Mithun George Jacob

A THESIS

in

Robotics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Master in Science in Engineering

2009

_____

Jianbo Shi
Supervisor of Thesis

_____

Jianbo Shi
Graduate Group Chairperson

Table of Contents

List of Figures

Acknowledgements

I'd like to thank my advisor Prof. Jianbo Shi for his constructive feedback and all the advice he has extended. I'd also like to thank Prof. Kostas Daniilidis for his guidance with the visual odometry module and Jack Sim for all the data and help he provided as well as Roy Anati for the useful talks we've had.

Also, I thank my parents for their never-ending support, prayers and their eternal confidence. Special thanks goes to Ria for labeling 18 images out of the 100 image dataset: every image counts! And also thanks to all my friends who provided moral support in their own ways over the last few months.

And finally, to all the unknown people I probably inconvenienced by converting the computer lab in the Moore building to my own personal cluster from midnight to 8AM, my thanks and apologies!

Abstract

Semantic segmentation of driving scene videos is fast becoming an important research topic with applications ranging for AGVs to driver safety. A system is proposed to perform object recognition and segmentation on a sequence of images capture from a completely calibrated stereo rig by only using simple structure-from-motion cues.

We first use the images and the calibration matrices to robustly and accurately determine the frame-to-frame motion which allow us to compute projection matrices for cameras in all the frames in the driving sequence. Then we proceed to obtain lower-quality correspondences to create a trajectory table of interest points, tracking them across the whole sequence of images. Using this trajectory table and the projection matrices, we create a sparse, noisy 3D reconstructed world which contains world points corresponding to every interest point in the image.

We also created a labeled database of 100 images and trained a randomized decision forest on it using simple structure-from-motion cues and conducted experiments which demonstrate that the system works well and performs fairly even in the face of pathological input.

COPYRIGHT

Mithun George Jacob

2009

Chapter 1

Introduction

Segmentation and recognition in images is one of the most important functions computer vision can offer an autonomous system. In this work, we investigate how an image can be accurately segmented and recognized using sparse, noisy 3D point clouds obtained by structure from ego-motion. (SfM).

Brostow et al. [1] demonstrated that it is possible to achieve this for driving videos with the aid of five simple cues derived from the SfM point clouds which are projected to the image plane. These cues are then used in conjunction with rectangular features to semantically segment the images using a randomized decision forest [5].

The experiments conducted by Brostow et al. resulted in the creation of the CamVid Database [2] which was obtained from HD footage procured by a car with a camera mounted inside. In addition to this footage, they have also provided 700 images with per-pixel ground truth for multiple classes as well as the intrinsic parameters of the camera. Using a commercial software [3], the authors also computed the extrinsic parameters of

the monocular sequence and were thus able to create sparse, noisy point clouds of the area traversed by the car.

In this work, we use the data generously provided by authors [2] as well as footage collected from Little Ben [4] of the Ben Franklin Racing Team (University of Pennsylvania, Lehigh University, Lockheed Martin) raced at the DARPA 2007 Urban Grand Challenge. Little Ben is an enhanced Toyota Prius hybrid car and is equipped with LADAR, radar sensors and most importantly (for the purpose of this work), stereo cameras.

The fully calibrated stereo rig on Little Ben is used to estimate frame-to-frame motion in the obtained footage and thus create the 3D point clouds representing the footage we are trying to segment (Chapter 2). Using the techniques described in [1] we first attempt to reproduce the results presented on the authors' data and proceed to implement it on our data (Chapter 3). The results of our experiments conducted on our data is presented in Chapter 4 and conclusions and future work are examined in Chapter 5.

Chapter 2

Structure from Motion Point Clouds

In order to obtain the motion and structure cues to enable semantic segmentation, we first need sparse 3D point clouds representing various points in the areas captured by the footage. The footage captured by the stereo cameras on Little Ben provide us with a sequence of stereo pairs capturing the scene around the car along the route taken. This chapter discusses the techniques used to estimate the frame-to-frame ego-motion which will be used to determine the projection matrices of the camera at the point where each frame was captured.

Standard structure from ego-motion techniques rely on tracking 2D image features using an interest point detector such as Harris-Stephens corners [6] or SIFT detectors [7]. Using the point correspondences between multiple views of a scene, a trajectory track is built from the landmarks and using the most temporally separated frames and using the computed projection matrices we reconstruct the 3D locations of the landmarks observed along the route.

2.1 High Quality Landmark Detection and Tracking

SIFT [7] has been proven to provide robust image matches in the context of SLAM [8,9] and therefore we use SIFT interest points to detect landmarks. We utilize the Vedaldi-Fulkerson SIFT implementation [10] to obtain the SIFT feature descriptors of interest points in each frame. Additionally, landmark tracking was also achieved by SIFT matching between frames by assigning a score to each match where the score was computed as the squared difference between the feature descriptors.

This method has been proposed [11] to be better than other tracking techniques such as a KLT-based tracker [12] for a similar driving scene application. Traditionally, a match is accepted if the ratio between its score and the second-best match's score is greater than a predetermined threshold. We find that a ratio of 1.5 is quite adequate and provides sufficient interest points for accurate ego-motion estimation. Since we utilized a completely calibrated stereo rig, we rectified the images [13] and used this to further eliminate landmarks if the distance between the height of the projected landmark on the stereo set was below 4 pixels (i.e. the distance of each point from the corresponding epipolar line). In order to further improve the quality of the matches, interest points which appeared in all 4 images (from the two sets of stereo images) are only selected as landmarks.

2.1.1 Tracking features through the stereo pairs

Features are tracked through the four images in the 2 stereo pairs by individually

matching 3 images to an arbitrarily chosen 4th image (the right camera image of the second stereo pair) and points in the 4th image which are found to be present in the other 3 images are retained.

This results in the question, why match 3 images to one? It would much cheaper (computationally) to match points from say, image 1 to image 4 and then match those points to image 2 and then to image 3 reducing the number of matched points dramatically each time.

Now, assuming we did this, suppose there exists a point in image 4, P_4 which actually matched to 2 points in image 1 and since the SIFT score ratio ties are arbitrarily broken, suppose P_1 was selected and the other point was P'_1. Now, when matching points from image 1 to image 2, it is definitely possible (and has been observed) that P'_1 is actually a better match than P_1. But since only P_1 is available and since it's SIFT descriptor is sufficiently similar to that of P'_1 it will definitely be matched thus resulting in suboptimal matching in the feature track along the 4 images.

But if there were no discarding of points as we go along and individually match all the points in 3 images to the points in a single image, only the optimal matches will be retained and if the optimal matches includes all 3 images, then the point is retained – which is the desired result ensuring high quality feature tracking.

2.2 Pose Estimation

Once robust correspondences have been determined, there are several available techniques to estimate inter-frame ego-motion within a hypothesize-and-test architecture. This architecture works by first generating a hypothesis of the ego-motion using a subsample with the minimum number of correspondences required to generate the hypothesis and then to test it on all the correspondences.

We experimented with several hypothesis generators such as the normalized 8-point algorithm [14] which uses 2D-2D correspondences between frames captured by the left or right camera. Another technique of interest was the 5-point method [15] which uses 2D-3D correspondences and was utilized in [16] where a 3D cloud is created using the stereo pair and feature points from the image in the next pair is used to estimate the relative pose.

The algorithm developed in this work utilizes 3D-3D correspondences obtained from matched stereo point clouds from the rig at different time steps. Since the stereo rig is completely calibrated, each stereo cloud is reconstructed using two-view DLT triangulation [14]. Using Horn's absolute orientation solution [17] (provided here for completeness), and working under a rigid-body motion assumption (since we can assume that ego-motion is the dominant cause for pixel flow) we are able to accurately determine frame-to-frame motion. Additionally, in order to obtain an optimal solution, 3D-3D correspondences whose reprojection error on their respective image planes is less than 2

pixels are used.

2.2.1 Closed-form Solution of absolute orientation

Given *n* 3D-3D correspondences between two 3D point clouds, the objective is to find the relative motion between them, i.e .assuming rigid body motion, the rotation and translation. Let each point cloud be represented by r_1 and r_2 and therefore, we need to find the rotation, *R* and translation *T* such that we minimize the residual error between each correspondence,

$e_i = r_{1,i} - R(r_{2,i}) - T$ where $r_{j,i}$ represents the i^{th} correspondence in the j^{th} cloud

The error minimized is the sum of the squared residual error for each correspondence,

$$\sum_{i=1}^{n} \|e_i\|^2$$

Let us translate each point cloud such that the centroid serves as the origin and let these translated point clouds be referred to as R'_1 and R'_2 . Using the 3x3 matrix *M*,

$$M = \sum_{n}^{i=1} R'_{1,i} R'^{T}_{2,i}$$

whose elements are the sums of the products of the coordinates in the first cloud with the coordinates in the second cloud. As stated by Horn, this matrix contains all the information required to determine the rotation and subsequently the translation between the point clouds. Representing M as:

$$M = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}$$ and then let N,

$$N = \begin{bmatrix} (S_{xx}+S_{yy}+S_{zz}) & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & (-S_{xx}+S_{yy}+S_{zz}) & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{bmatrix}$$

As shown in [17], the rotation which best minimizes the aforementioned error is obtained by the unit quaternion which is the eigenvector corresponding to the most positive eigenvalue of *N*. The quaternion is converted to it's corresponding rotation matrix *R* and this is used to obtain the optimal (in a least-square sense) translation, T.

$T = r_{1,m} - R(r_{2,m})$ where $r_{j,m}$ represents the centroid of cloud r_j

Therefore, using this algorithm it is possible to estimate the motion between two 3D point clouds containing at minimum 4 correspondences [17].

2.2.2 Hypothesize-and-test

The hypothesize-and-test architecture is summarized in Algorithm I.

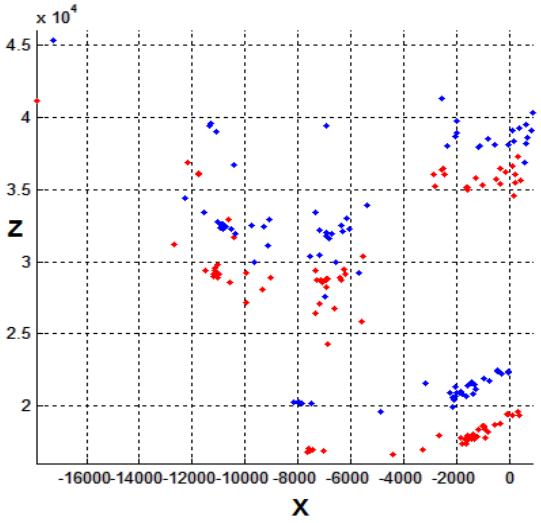
Algorithm I
Input: *n* 3D-3D correspondences between two stereo point clouds
Output: A rotation *R*, and translation *T* minimizing the sum of residual errors

Main

1. Randomly select 4 3D-3D correspondences
2. Compute *R, T* and the error (in millimetres)
3. Determine the number of inliers, $n_{inliers}$
4. $$L = \frac{\log(1-0.999)}{\log\left(1-\left(\frac{inliers}{n}\right)^4\right)}$$
5. If total number of trials to date is greater than *L* or 3000, return *R* and *T* with the least error

Figure 1. Algorithm I - Pose Estimation

Figure 2. Robust correspondences across the stereo pairs. *Top Row* – Stereo pair ; *Middle Row* – Stereo pair after 3 frames ; *Bottom Row* – 2D view of 3D-3D correspondences (Red points – Before movement ; Blue points – after movement)



We use a RANSAC -based [18] estimation procedure to determine the best rotation and translation. Inliers are classified as correspondences which exhibit an error of less than 500 mm and the estimation is set terminate at most after 3000 trials.

2.3 Projection Matrix Computation

Using Algorithm I, we can robustly determine the motion between two frames. Now, we need to find the projection matrices of each camera (left and right) at each frame. We assign the camera center of the left camera of the first frame in the video as the origin. Since the stereo rig is completely calibrated, obtaining the projection matrix of the right camera is trivial given the projection matrix of the left camera. We obtain the rotation matrix and translation of the entire sequence by continually multiplying the inter-frame rotation and adding the translation as shown in Algorithm II.

Algorithm II
Input: Inter-frame motion between *n*-1 frames represented by $R_2, R_3, \ldots$ and $T_2, T_3, \ldots$ and $R_1 = I$ and $C_1 = [0\,0\,0]^T$
Output: Projection matrices for *n* frames, *P* where $P_1 = K\,I\,[I|-C_1]$
Main

1. $R' = R_1$
2. for i = 2 → n
   a. $R' = R_i * R'$
   b. Use SVD to find *U* and *V* such that $R' = U * S * V^T$
   c. $R' = U * V^T$
   d. $C_i = C_{i-1} - inv(R') * T_i$
   e. $P_i = K\,R'[I|-C_i]$

Figure 3. Algorithm II – Projection Matrix Computation

Due to the repeated multiplication of the rotation matrix, it is observed that it ceases to be truly orthogonal. Therefore, the rotation matrix is re-orthogonalized using singular value decomposition to obtain the closest orthogonal matrix to the rotation matrix in a

Frobenius-norm sense. In Figure 1, we display motion after 4 frames but in reality, we calculate motion between every frame.

Hence, using Algorithms I and II we can compute the projection matrix of the camera at each frame using the camera center of the left camera of the first frame as the origin. As we will see in Chapter 3, these projection matrices will be used to compute the SfM point clouds to be used in semantic segmentation.

Chapter 3

Semantic Segmentation with Point Clouds

In Chapter 2, we discussed the algorithms used to obtain the projection matrices of the camera at every frame. In this chapter, we discuss how five cues are extracted from the point clouds and utilized for the semantic segmentation of the image.

3.1 Feature Tracks and Triangulation

During the SIFT interest point detection phase in Chapter 2, we only extracted high quality correspondences since we only needed 4 good correspondences to accurately determine inter-frame motion. But for segmentation, it is important to extract as many reasonably good correspondences as possible since coverage of the image is crucial. This is achieved by lowering the SIFT matching score ratio to 1.1 and increasing the epipolar distance threshold between stereo correspondences to 6 pixels. An interest point is still only accepted if it is tracked through all 4 images (two stereo pairs) even though this results in the rejection of several points (this will be re-examined in Chapter 5).

Once all the points have been collected, a trajectory table is generated which keeps track of each point in all the frames in the sequence by assigning a unique ID to it. The other columns store information such as the frame the point is visible in, and its pixel coordinates in that frame.

3.1.1 Unique ID Conflict Resolution

In the creation of this trajectory table, interest points were matched between 2 stereo pairs with the method discussed in Chapter 2 where we discussed how an interest point would only be accepted if it were tracked along all 4 images. This was achieved by comparing 3 images to a 4^{th} reference image (arbitrarily chosen). But this produces an interesting problem when generating unique IDs for interest points.

Suppose a point P_D in the reference image D is successfully matched to corresponding points in images A, B and C where A-B is a stereo pair and C-D is the other stereo pair and is assigned an ID X. But suppose there was another point P'_C which lost out on the previously mentioned arbitrary tie-breaking. Now, in the second iteration we have C-D and E-F as the stereo pairs but it happens that P_F matches to P_D and P'_C and not to P_C. This will result in the assignment of the same ID X since the common reference point P_D (since we have arbitrarily chosen the right camera of the second stereo pair as the reference image) possesses that ID.

Again, these "ties" for the ID are arbitrarily broken by choosing the first instance. Since these do not happen often and the as expected, the pixel positions between these

points are quite small, discarding the other points does not affect performance.

Once the trajectory table for each point is created, we identify the most temporally separated frames providing a wide baseline for the 3D reconstruction and saves needless computations [1]. Since the projection matrices are created with the camera center of the left camera of the first frame as the origin, each point in the video sequence is assigned **one 3D point in the world coordinate system**. This property will be examined later in this chapter.

3.2 Motion and Structure Cues

Using the one world 3D point, W generated for each interest point in each frame, we use this information to extract five motion and structure cues [1] from each 3D point which will be used for semantic segmentation (reproduced here for completeness).

3.2.1 Height above the camera, f_H

Since the height of an object remains fixed in a normal driving sequence (unless one crosses an elevated object like a bridge which is not bound to happen frequently), the height of the 3D point from the camera is used such that $f_H = W_y - C_y$ where C denotes the position of the camera center in that frame in the world coordinate system.

3.2.2 Closest distance to camera path, f_C

Again, since several objects like buildings, trees, the sidewalk are separated from the camera center (i.e. the car) by a fixed range of values, the closest distance of the 3D point to the entire camera trajectory (i.e. the route of the vehicle) is used. Hence, using the entire sequence of camera centers $C(t)$, $f_C(W) = min_t \|W - C(t)\|$

3.2.3 Backprojection Residual, f_R

The backprojection residual measures the rigid-world assumption of the world point since every interest point has **one** world point, the reprojection error would be different for all the frames in between (the most temporally separated frames) if the object was mobile. Therefore, by tracking the points along several frames and reprojecting one world point to each frame, we have a much better estimate of the reprojection error as opposed to projecting the 3D point obtained from the stereo rig (which would always be minimal). Logarithm scaling, $f_R(W)=\log(1+q(W))$ is used [1] to ensure that the residuals from mobile points is not dominated by tracking errors on distant objects (like the sky). $q(W)$ is the 2D variance of the reprojection error of the world point *W* with respect to its trajectory track in pixels.

3.2.4 Track Density, f_D

Track density is another useful cue since it can be used to distinguish feature-rich regions like trees and buildings from smooth regions like the sky or the road.

3.2.5 Surface Orientation, f_{Ox}, f_{O_y}

Since the reconstructed point clouds are sparse and inaccurate, creation of a faceted world is not desirable. Instead we use the 2D connectivity between the interest points in the image to obtain a Delaunay triangulation and the normal of the corresponding 3D triangle in the world coordinate system is projected onto the image plane such that the *x* and *y* components of the normal is f_{O_x} and f_{O_y} which provides a heuristically acceptable local surface orientation.

3.3 Features for Classification

Rectangle features have been successfully used in segmentation [20,19,1] and are used here as well. For every pixel in the image, a rectangle of predetermined size and offset from the pixel is used. If we use an optimal rectangle, we will be able to extract contextual information with respect to the pixel (such as sidewalks adjacent to roads, etc.) and finding this "optimal" rectangle will be explained in detail in Section 3.4.

By defining a rectangle relative to each pixel, we implicitly define a truncated pyramid in the world coordinate system. Therefore, by summing up cue values present in the rectangle (and thus the pyramid) we try to capture contextual information.

For fast feature response computation, each frame was converted into a sparse representation of points (a feature image) where the location of the point corresponded to pixel coordinates of the projected landmark and the value associated with each point would be the value of the feature response for its corresponding world point. Therefore, we generated 6 feature images for each frame, the first three containing values equal to f_H, f_C, and f_R for the corresponding world point. The track density was encoded as points with value 1.

Since the surface normals cover the entire image and not just the locations of the landmarks, they were generated separately and each pixel takes on the value of the normal of *x* and *y* component of the projected 3D triangle in two feature images (one for

f_{O_x} and f_{O_y}). Integral images [20] were used to rapidly sum up the cue values in each rectangle by creating an integral image for each cue (by using the feature images).

Once the summed up cue values are computed for each pixel, these 6 values form the feature vector which will be used for classification.

3.4 Learning for Semantic Segmentation

In the previous section, we discussed the computation of the feature vector of each pixel in the image using offset rectangles, but neglected to mention how these rectangles are determined. The next subsection will serve as an introduction to randomized decision forests which are the classifiers used in this work and their application.

3.4.1 Randomized Decision Forests

Randomized decision forests have been used for a variety of classification and regression tasks and overcoming the overfitting characteristic of classic decision trees like ID3 or C4.5 by introducing randomness into the splitting functions. The randomized decision trees used here generate 500 random splitting functions at each level defined by

- a cue selection from 1 – 6
- a threshold value in $[0,1]$
- a rectangle whose offset lies in $\left[\frac{-w}{2}, \frac{-w}{2}\right]$ and $\left[\frac{-h}{2}, \frac{-h}{2}\right]$ and size lies in $[0,w]$ and $[0,h]$ where $\{w, h\}$ = width, and height of the image

Once 500 splitting functions have been generated, they are applied to all the pixels to be classified in all the images of the training set and the splitting function which separates

the classes best is chosen and stored. The set of pixels is then split into two sets which then recursively split again by generating and evaluating a new set of 500 random splitting functions until the number of remaining pixels left to be classified falls below a threshold, the tree has reached a particular depth or all the remaining pixels belong to a single class. When this happens, the leaf is set as the class occurring at the highest frequency in the remaining pixels.

The splitting function provides 3 crucial pieces of information required to generate a feature response for a pixel: a rectangle, choice of cue and a threshold value. The rectangle defines the set of projected landmarks (interest points) to be used in feature computation. The cue choice defines which cue values to sum up in the rectangle and thus generates the feature response per pixel.

Suppose all the feature response values were stored in a vector *V*, and the random threshold value in $[0,1]$ was *s,* and the splitting threshold value *T* is defined as follows, $T = min(V) + (max(V) - min(V)) * s$, then the splitting function would separate the set of pixels, *S* into two subsets such that the pixels whose feature responses were less than *T* would be classified by the left child splitting function (to be evaluated recursively) and the other pixels would be classified by the right child splitting function.

3.4.2 Normalized Information Gain

In the previous section, we discussed how 500 splitting functions were randomly generated and the function which best separated the classes was chosen. The "best"

separation score used here is the normalized information gain [21] which normalizes the mutual information gain such that it lies between 0 and 1. The score is defined as follows,

$$Score_C(T) = \frac{2\mathrm{I}_C^T(S)}{H_C(S) + H_T(S)}$$

where T is defined as the splitting test, $H_C(s)$ is the prior classification entropy in the set of pixels S, $H_t(S)$ is the entropy of S with respect to the test T and $I_C^T(S)$ is the information provided by the splitting function on the classification of S. Therefore, a higher value of the score implies a better separation of classes within S.

3.4.3 Classification

Once a set of trees are trained on a set of images, they are aggregated together to form a randomized decision forest. Each tree votes towards the classification of the pixel and the majority vote stands. This aids in the robust segmentation of pixels and reduces overfitting on the training data. It is also proven that prediction error is a monotonically decreasing function of the number of trees in the randomized forest [22] and therefore an adequate number of trees have been grown for classification (the details of which will be discussed in the next chapter).

Once the decision tree is generated, classifying a set of pixels is similar to the training procedure where the pixels are recursively split using the splitting function encoded at each node until a leaf node is reached where all the nodes are classified as the class at the leaf.

Chapter 4

Experiments

Using the algorithms described in the previous chapters, we implement the semantic segmentation system first on the data provided by the authors of [2] and then on the data from the data collected by Little Ben [4].

4.1 CamVid Database

As mentioned before, we use the provided projection matrices, labeled images and trajectory table to train a randomized decision forest of 50 trees trained to a depth of 11 and the minimum sample size of 3000 pixels. A training set of 60 images of size 720x960 pixels was chosen and since it was not possible load all 60 images into memory given the computational resources at hand and loading the images from file would dramatically slow down the process, the training set was randomly subsampled into 50 sets of 6 images such that each image was uniformly covered by the 50 training sets (for each tree in the forest).

Even with the subsample of 6 images, it took approximately 3.5 hours to grow one

randomized decision tree. Given the fact that 500 decision functions are generated to decide each node of the tree and each tree contains approximately 820 nodes, and that classification at the root begins with 720x960x6 pixels (i.e. 4147200 pixels), this implies that it took approximately 30ms to evaluate each of the 500 decision functions. This corresponds with the profiling done during the code optimization process before batch processing all the images.

In order to grow a forest in a reasonable amount of time, a cluster of 25 computers (Intel Core2Duo E8400, 3.00 GHz and 3.25 GB RAM) was used for training and classification. Classification took only 210s per frame with the help of the Parallel Processing Toolbox in MATLAB (which saved 30s per frame).

Since it was apparent that [1] the best results obtained from semantic segmentation using SfM point clouds would result in an accurate segmentation of major classes present in driving scene videos such as the road, buildings, trees, the sidewalk and the sky, a shallower tree and a comparatively high minimum population was chosen to obtain a smoother result.

In Figure 4(a), we display the results of using simple ego-motion-derived cues to perform object recognition and segmentation using the CamVid database as well as the proportion of features used in the learned randomized segmentation forest in Figure 4(b).

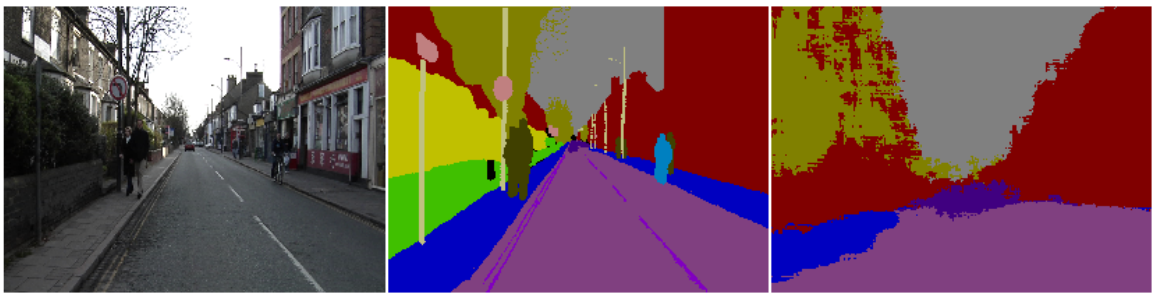


Figure 4(a) Testing Set: Raw image – Labeled truth image – Segmented image.

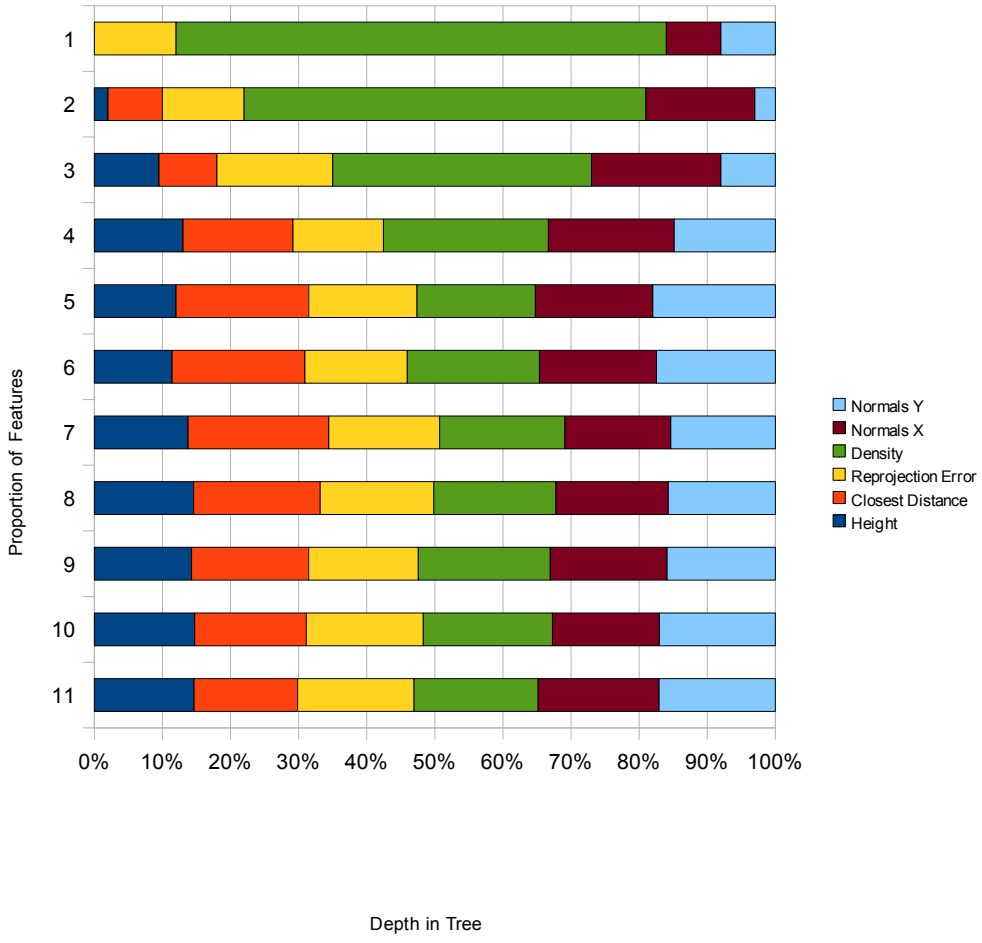


Figure 4(b) Proportion of features as a function of node depth in the randomized decision forest trained on the images in the CamVid Database

In correspondence with the results from [1], there is a similar but not exact bias towards density, reprojection error, height and closest distance near the roots of the tree but as we work our way to the leaves, all the cues work together to segment the finer details after a coarse segmentation by the cues popular near the root.

Additionally, in the test sequence we chose, we obtained a global accuracy of 64% which is close to that reported by [1], i.e. 61.8 %.

4.2 Little Ben footage

Since the only information provided were the stereo images captured by Little Ben and the intrinsic and extrinsic parameters of the stereo rig, we needed to generate the projection matrices, trajectory table and labeled images required for segmentation.

Projection matrices were computed using the algorithms described in Chapter 2 and for demonstrative purposes the sequence of camera centers has been overlaid on the corresponding map from Google Maps by using the turning points to determine the scale of the map and the starting positions (more on this in Chapter 5). The sequence of camera centers for two 600 frame sequences are shown in Figure 5.

In order to accurately train and test this method on this footage, 100 images were hand-labeled and as before, 60 were used for training but 40 were used for testing. The test set was carefully chosen so that include cross-street scenes which are not present in

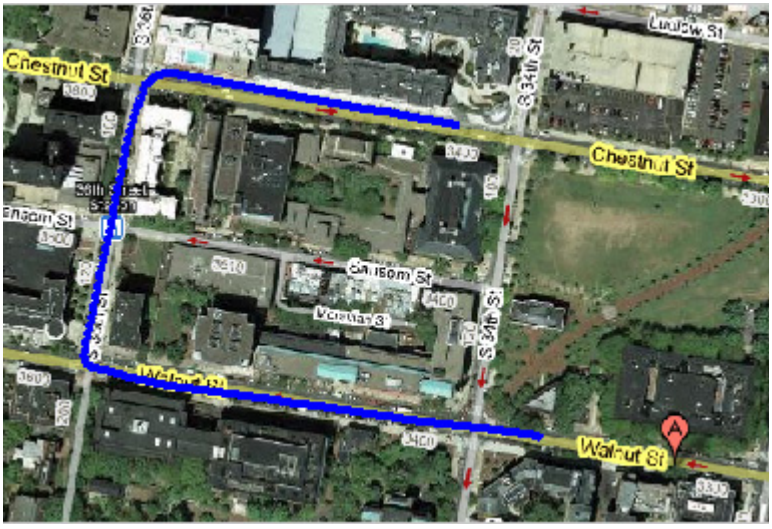


Figure 5(a) Visual Odometry: Sequence 1600 – 2200 ~ 0.4 miles

Figure 5(b) Visual Odometry: Sequence 14260 – 14860 ~ 0.3 miles

the training set and as expected performance deteriorated but not dramatically. The classification performance of the pathological test frames are enclosed in the rectangle shown in Figure 7.

The final classification result on one of the test frame is shown in Figure 6. The mean global classification accuracy is 51.45% and global training accuracy is 76.81%. If we consider the non-pathological test images in the test sequence, i.e. 40%, we obtain results comparable to that of [1] and our prior results on the CamVid Database.

A visual representation of a decision tree from the CamVid and Little Ben forests is displayed in Figure 8 to show that the both trees in the upper levels are fairly balanced trees as is expected from good splits but as we work our way down to the lower levels, the splits start to become skewed.

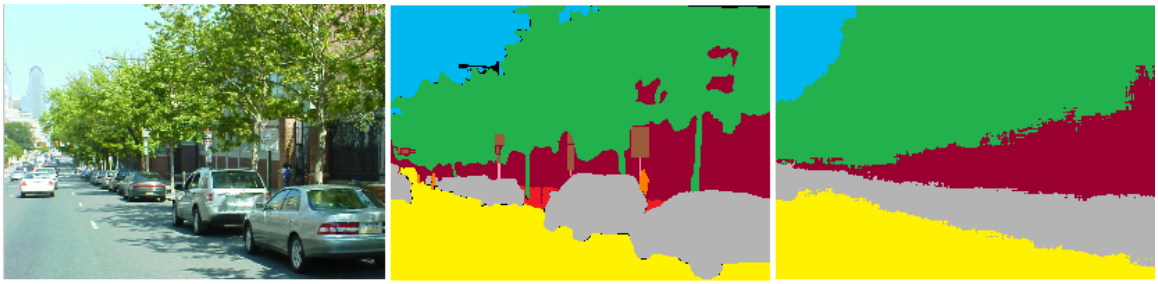


Figure 6(a). Testing Set: Raw Image – Labeled Image – Segmented Image

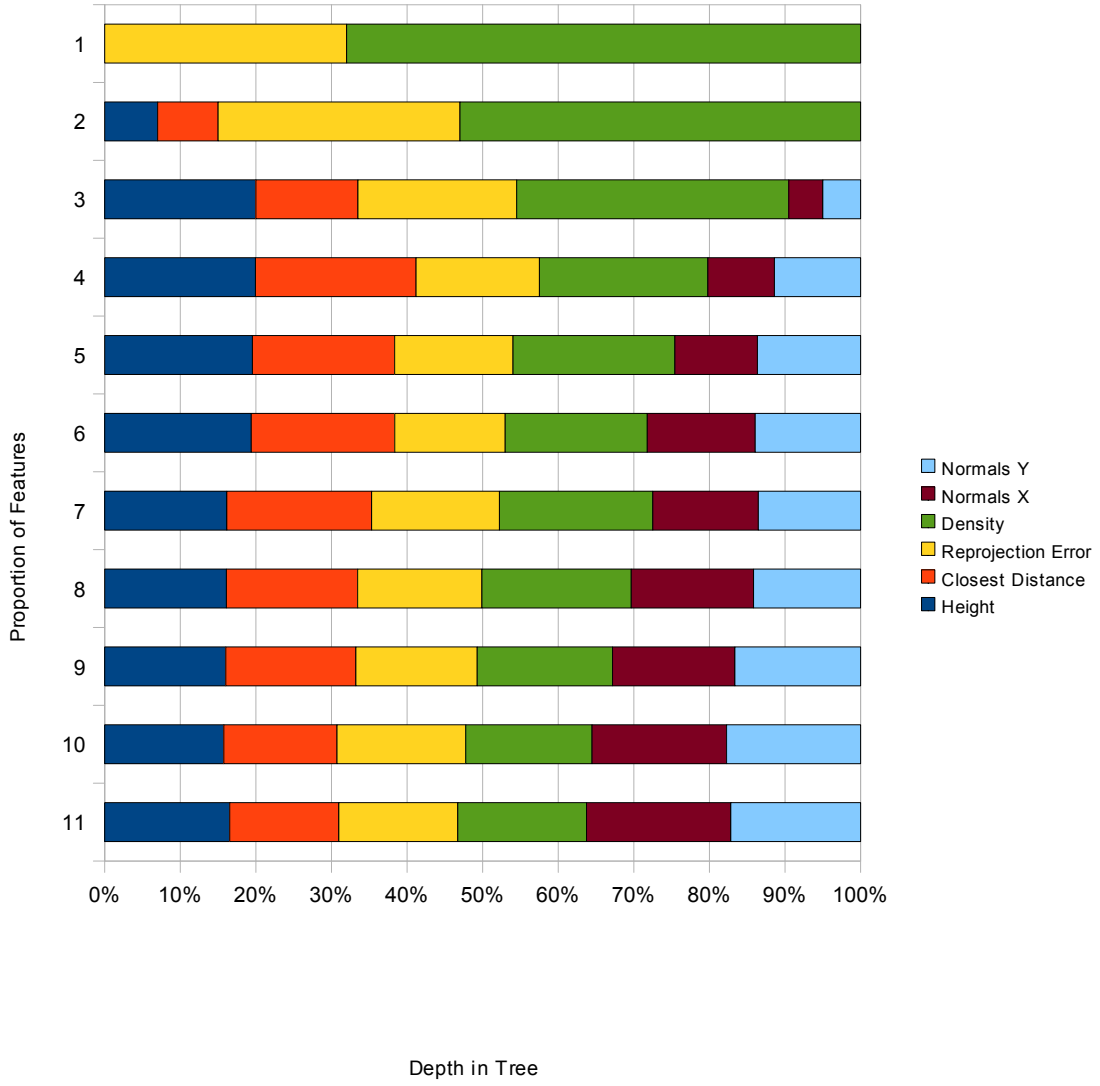


Figure 6(b) Proportion of features as a function of node depth in the randomized decision forest trained on the images in the Little Ben footage

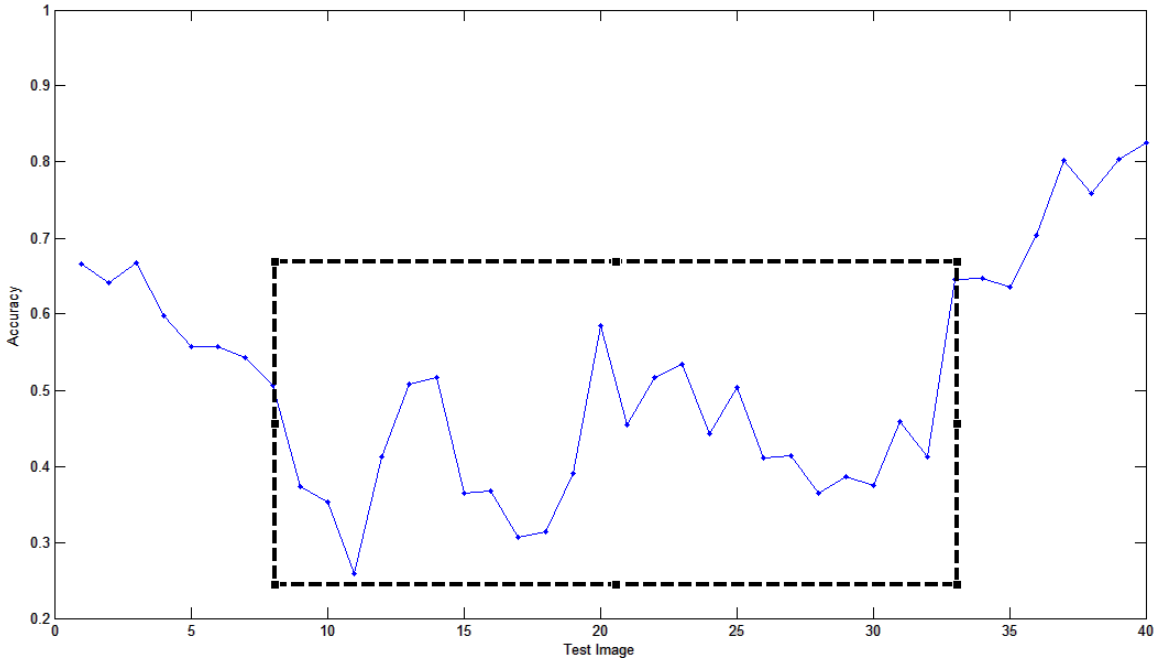


Figure 7. Test Accuracy (Little Ben footage(

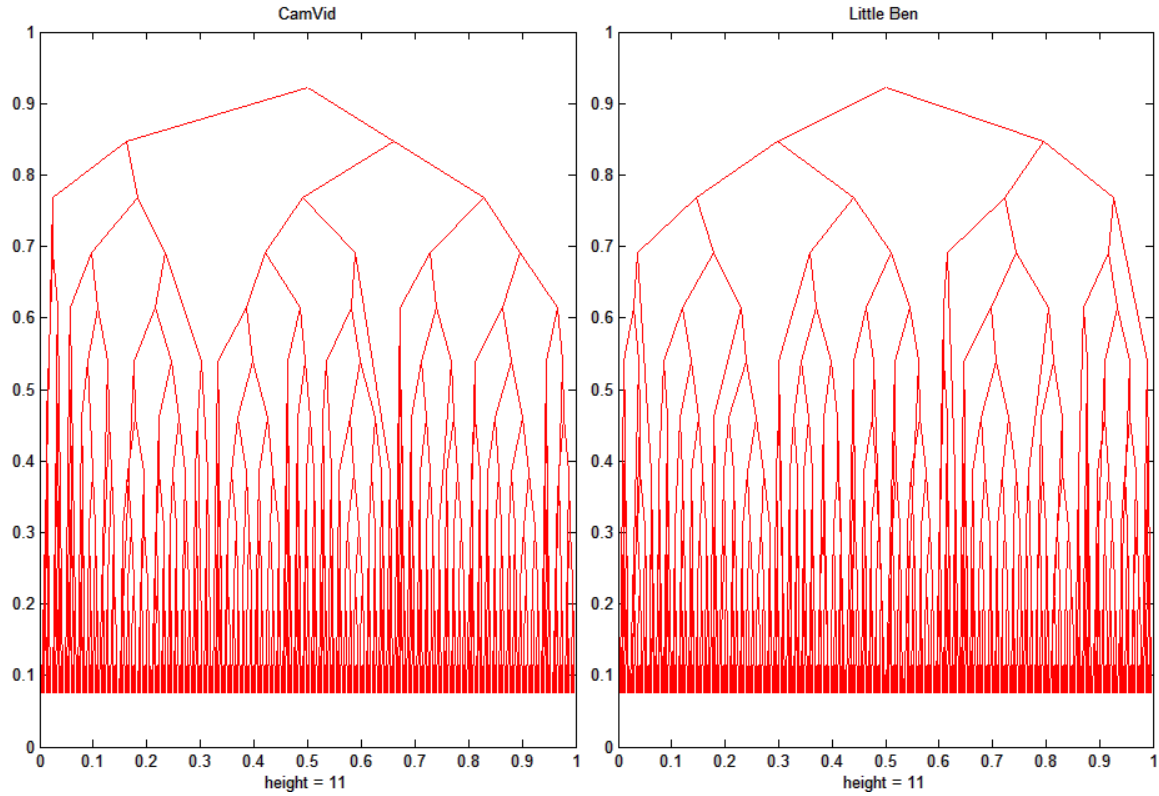


Figure 8. CamVid and Little Ben randomized decision trees

Chapter 5

Conclusions

We have implemented a system to perform object segmentation and recognition on images with the aid of simple ego-motion derived cues. Additionally, we have developed a robust stereo visual odometry system capable of computing projection matrices of the camera at various time steps throughout the route.

The segmentation system utilizes the projection matrices to create a 3D world where each interest point in the video sequence is represented by a single 3D point. Using the various cues extracted with rectangle features over the image, a randomized decision forest is grown by testing 500 splitting functions at each node.

The system was tested on the CamVid Database and on a new driving video database created from the stereo images obtained from the Little Ben footage.

Upon analysis of the results, it is apparent that since driving scenes are limited to a limited set of scenarios, we can improve the performance of the classifier by using a

training set with several images depicting as many different scenarios as possible.

Additionally, it has been shown that the system can be used to successfully segment the image coarsely. This knowledge can be used to locally segment regions and simplify the the overall task of semantic segmentation of driving scenes.

Also, the visual odometry system assumes that the 3D-3D correspondences it obtains are high-quality matches and also that dominant pixel flow is due to ego-motion. But certain scenarios can result in these assumptions being grossly incorrect. For example, if faced with a feature-less wall, the correspondence quality dramatically drops. Also, and more importantly, if there is a large vehicle such as a bus, with feature-rich rears blocking the camera view, then the second assumption is also rendered invalid.

These problems can be tackled by looking for planes of a specific orientation, i.e. parallel to the image plane in the world coordinate system. Since only the rears of vehicles and walls can offer this kind of geometry, it may be advisable to ignore correspondences obtained on these objects.

Another point of interest is the total lack of the use of appearance based features. Integrating such features as a post-processing step given the knowledge of the general layout of the scene, would perhaps increase the classification performance as compared to just using appearance based features.

Also, it would be most interesting to create a forest using the whole labeled set of 100 images and observe the classification results.

Since the visual odometry module of the system can be extended to be more robust, more work is definitely possible there. At the moment, the overlay on Google Maps is quite naïve, but given the starting longitude and latitude of the vehicle, it maybe be possible to accurately determine the scale and starting position with this information instead of finding the start position through trial and error.

And finally, different values of the minimum population and height of the tree can be investigated to determine which set of parameters produce the best possible decision forest.

References

1. Gabriel J. Brostow , Jamie Shotton , Julien Fauqueur , Roberto Cipolla, "Segmentation and Recognition Using Structure from Motion Point Clouds", Proceedings of the 10th European Conference on Computer Vision: Part I, October 12-18, 2008, Marseille, France
2. Gabriel J. Brostow, Julien Fauqueur, Roberto Cipolla, "Semantic object classes in video: A high-definition ground truth database", Pattern Recognition Letters, Volume 30, Issue 2, Video-based Object and Event Analysis, 15 January 2009, Pages 88-97, ISSN 0167-8655
3. Boujou: 2d3 Ltd., http://www.2d3.com (2007)
4. Ben Franklin Racing Team, http://www.benfranklinracingteam.org (2007)
5. Geurts, P., Ernst, D., Wehenkel, L., "Extremely randomized trees", Machine Learning **36**(1) (2006) 3-42
6. Harris, C., Stephens, M., "A Combined Corner and Edge Detector" In: 4th ALVEY Vision Conference. (1988) 147 – 151
7. Lowe, David G., "Object recognition from local scale-invariant features", Proceedings of the International Conference on Computer Vision. 2. pp. 1150–1157 (1999)

8. T. D. Barfoot, “Online visual motion estimation using fastslam with sift features”, Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, pages 579–585, 2005.
9. P. Elinas, R. Sim, and J. J. Little. “sslam: Stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution”, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), page 1564–1570, 2006.
10. A. Vedaldi and B. Fulkerson. “VLFeat: An Open and Portable Library of Computer Vision Algorithms”, http://www.vlfeat.org/, 2008
11. Jean-Philippe Tardif, Yanis Pavlidis, Kostas Daniilidis, “Monocular visual odometry in urban environments using an omnidirectional camera”, IROS 2008: 2531-2538
12. B.D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereo Vision” In Int. Joint Conf. on Artificial Intelligence, pages 674–679, 1981.
13. J.-Y. Bouguet, Camera Calibration Toolbox for Matlab, Supplied by Computer Vision Research Group, Department of Electrical Engineering, California Institute of Technology.
14. Hartley, R.I., Zisserman, A, “Multiple View Geometry in Computer Vision”, Second edn. Cambridge University Press (2004)
15. D. Nister. “An efficient solution to the five-point relative pose problem”. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26:756–770, 2004.
16. Z. Zhu, T. Oskiper, O. Naroditsky, S. Samarasekera, H. S.Sawhney, and R.

Kumar. “An improved stereo-based visual odometry system”, USA, August 2006.

17. B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions", J. Opt. Soc. Amer. Avol. 4, no. 4, pp. 629-642, Apr. 1987.
18. M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24:381–395, 1981.
19. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: ECCV. (2006)
20. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR. (2001) 511-518 vol.1
21. Wehenkel, L., & Pavella, M. (1991). Decision trees and transient stability of electric power systems. Automatica, 27:1, 115–134.
22. Breiman, L. “Random Forests”, Machine Learning, 45, 5 – 32 (2001)