DYNAMIC SCENE INTERPRETATION AND UNDERSTANDING FROM TWO VIEWS

by

NINAD SHASHIKANT THAKOOR

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2009

To my Aai-Baba, family and friends.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Jean Gao for guiding me onto the right path for research. Her constant encouragement and motivation were pivotal in my research.

I am grateful to my co-advisor Dr. Venkat Devarajan for his guidance and support during my doctoral studies. I would like to thank Dr. Sungyong Jung, Dr. Qilian Liang and Dr. Michael Manry for devoting their time to serve on my dissertation committee.

I greatly appreciate the faculty and the staff at electrical engineering department and computer science and engineering department for bearing with me and for being more than helpful.

I would like to thank my present and former colleagues at the Biocomputing and Vision lab for making it the the best place to be at UTA.

I would also like to thank all my friends who made Arlington a home away from home for me starting day one till today.

However, none of this would be possible without my family, their encouragement and sacrifices which have gotten me here from a tiny village in India.

November 23, 2009

ABSTRACT

DYNAMIC SCENE INTERPRETATION AND UNDERSTANDING
FROM TWO VIEWS

Ninad Shashikant Thakoor, Ph.D.

The University of Texas at Arlington, 2009

Supervising Professor: Jean Gao

Co-Supervising Professor: Venkat Devarajan

Interpretation of a static or dynamic scene starts by segmenting the scene followed by recognition. Our work concentrates on the general problem of segmenting and recognizing animate and inanimate objects in a scene captured from two different views. The two views here refer to either a pair of frames captured by a stereo camera or two frames (with spatial overlap) captured with a moving camera.

The work described in the dissertation starts with an iterative split-and-merge framework for segmentation of an unknown number of objects captured with stereo camera. The disparity of a scene is modeled by approximating various surfaces in the scene to be planar. In the split phase, the number of planar surfaces along with the underlying plane parameters is assumed to be known from the initialization or from the previous merge phase. Based on these parameters, planar surfaces in the disparity image are labeled to minimize the residuals between the actual disparity and the modeled disparity. The labeled planar surfaces are separated into spatially continuous regions which are treated as candidates for the merging that follows. The

regions are merged together under a maximum variance constraint while maximizing the merged area. A multi-stage branch-and-bound algorithm is proposed to carry out this optimization efficiently.

For moving objects, a framework is proposed for two-view multiple structure-and-motion segmentation. This segmentation problem has three unknowns namely the memberships, corresponding fundamental matrices and the number of objects. To handle this otherwise recursive problem, hypotheses for fundamental matrices are generated through local sampling. Once the hypotheses are available, a combinatorial selection problem is formulated to optimize a model selection cost which takes into account the hypotheses likelihoods and the model complexity. An explicit model for the outliers is also added for a robust model selection. The model selection cost is minimized through a branch-and-bound procedure.

Followed by segmentation, object recognition was applied to understand the scene. The segmented objects lack exact boundaries; thus shape based recognition or classification will not perform well. We follow a more general approach of visual object recognition instead. Visual object recognition relies on spatial image features to identify the objects. The state of the art visual object recognition approaches use a visual bag-of-words to represent images. Bag-of-features is an orderless collection of invariantly detectable image patches. The approach discards spatial relationships between these patches and, gives objects, their context and the background clutter equal importance. In a modification to the original visual bag-of-words, separate representations for positively and negatively relevant image patches are formed. Improvements in the classification accuracies due to the separation are demonstrated through experimentation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Scene interpretation and understanding is at the heart of many computer vision applications such as video surveillance, video retrieval, navigation of mobile robots, intelligent environments and assistance technologies for visually impaired or elderly. This dissertation addresses key problems in the area of scene interpretation and understanding. In this chapter, a literature survey of the various topics related to our research is provided. The most recent work is identified for each of these topics. Additional references under each of the topics are provided in the later chapters when these topics are discussed in greater detail.

1.1 Segmenting Scene with Stereo Disparity

Planar surfaces are abundant in any manmade environment. For example, indoor images contain walls, floors, ceilings etc.; outdoor images contain sidewalks, roads, walls, roofs etc. Additionally, objects such as humans and vehicles can appear to be planar if they are observed from a distance. The ground plane is present in virtually every scene. Thus, segmentation of planar objects would greatly assist the automatic analysis of a dynamic or static scene and it can be carried out based on the depth information alone.

When an object is captured with a stereo camera, it appears shifted in one view compared to the another. This shift is known as disparity. In general, the disparity is inversely proportional to depth. The real time estimation of disparity for a stereo camera has become a reality due to the advances in general purpose

microprocessor and signal processor technology, which has resulted in higher data handling and computation speeds and better architectures allowing pipelining and parallel processing. Small vision system by SRI International [1], Stereo-on-a-Chip system by Videre design [2] and DeepSea system by TYZX [3] are a few examples of commercially available real time stereo systems that can be used as depth sensors. While stereo vision based depth sensors are less accurate than conventional depth sensors such as laser radars (LADARs) and structured light scanners, stereo holds advantages of being cheap, portable, flexible, passive, quick, and power efficient. Therefore computer vision systems are increasingly integrating stereo vision into their framework.

The depth segmentation problem has been extensively dealt with in the terms of the range image segmentation. However, due to the different modality of the depth estimates achieved through stereo, a different treatment of the disparity segmentation problem becomes necessary. The disparity based spatial segmentation in which depth values drive the spatial segmentation has been addressed by some researchers [4]. Disparity and motion information are also combined to carry out segmentation [5,6]. However, very limited work has been done in segmentation of disparity alone. The following paragraphs summarize these efforts.

Se and Brady [7] apply random sample consensus (RANSAC) to the disparity values to detect the ground plane in their stereo vision based algorithm. They assume that the ground plane is always visible and is the dominant plane in the image. The detected ground plane is then used to detect any small obstacles against it. Okada *et al.* [8] utilize randomized Hough transform to the Euclidean 3D data calculated from the stereo. The peak points in the Hough space are selected as the plane candidates. Finally, each image point is associated with a closest plane candidate. Trucco *et al.* [9] apply their method to the disparity of a weakly calibrated stereo.

For each image pixel, the plane parameters are calculated by least squares fitting on the disparity values of the local neighborhood. The planes are found by carrying out clustering in the plane parameter space. However, for the point on the boundary of the planes, the calculated parameters are inaccurate. This leads to inaccurate plane labels in these regions.

Wang and Wang [10] use a planar surface model very similar to [9], the only difference is that range values are used instead of disparity. A Bayesian framework is used to impose a prior on the spatial continuity of the planar surfaces with a Markov random field (MRF). The range images are known to be "clean". Therefore, the improvement in the segmentation of range data due to the MRF formulation is not evident.

1.2 Segmenting Scene with Structure-and-Motion

The segmentation of structure-and-motion is a vital step towards interpretation of a dynamic scene. A typical dynamic scene structure includes multiple independently moving objects, which are being captured by a moving camera. Conventional approaches based on the frame difference [11, 12] or the 2-D flow based methods [13, 14] are restricted in segmenting such a scene. The frame difference based approaches are limited due to the need for camera motion compensation. The 2-D flow based approaches are limited by the typical affine camera model.

To address the problem in a better way, a comprehensive theory of structure-and-motion (SaM) estimation from perspective images has been developed by computer vision researchers over the years [15]. Analysis of dynamic scenes based on this theory, also known as multi-body structure-and-motion (MSaM) is now being extensively explored. A two-view MSaM segmentation problem starts with sparse image correspondences between two camera views and groups the correspondences with

same motion. Some of these correspondences are incorrect and should be treated as outliers. The two-view MSaM problem can be interpreted as a geometric problem [16] as well. However, its direct application to real world problems is limited as it does not account for the outliers. An interesting alternative is a clustering perspective. The two-view MSaM clustering is a chicken-and-egg problem. To segment the scene, one needs motion models for all the objects. To estimate the motion models of individual objects, one has to segment the objects first. To solve such a recursive problem, iterative techniques such as expectation maximization can be used [17]. However, results of expectation maximization can be guaranteed to be only locally optimal and hence depend on the initial motion models. An alternative to the iterative method is a sequential extraction strategy, where motion with largest number of matches, i.e. a dominant motion, is segmented and separated at a time until the entire scene is explained [18]. A limitation of such methods is that similar motions are often incorrectly segmented. The object encountered earlier in the search is assigned some fraction of other objects, which have similar motion.

To get out of the chicken-and-egg dilemma, some researchers have applied random sampling to generate multiple hypotheses for the motions in a scene [19, 20]. For the image correspondences, which occupy the same spatial neighborhood, the motion is expected to be the same. This knowledge of spatial coherency helps in the selection of reliable hypotheses by local sampling of image correspondences through a RANSAC-like process. Once hypotheses are available through sampling, a suitable cost function can be optimized to achieve motion segmentation. Another important aspect of clustering is the *number* of clusters. While most of the clustering techniques assume that the number of clusters is known, such assumption is invalid for the segmentation of a dynamic scene. Typically, clustering is carried out for a varying number of clusters and, the best according to a certain criterion is selected.

1.3 Visual Object Recognition

Object recognition is a classic problem in image processing and computer vision. Among others, object recognition based on shape is widely used [21–24]. A shape can be represented either by its contour or by its region [25, 26]. Curvature, chain codes, Fourier descriptors, etc. are contour based descriptors while medial axis transform, Zernike moments, etc. are region-based features. Contour based descriptors are used as they preserve the local information that is important in classification of complex shapes. However, medial axis or skeleton based descriptors such as shock graphs [23, 24] and bone graphs [27] are popular in vision community as well. The skeleton based descriptions preserve structural information for each part of the shape with relation to the others. Thus, in case of articulation, viewpoint change and, partial occlusion, skeleton based approaches are more successful. However, almost all shape based approaches require the extraction of exact boundaries of an object in a scene, which still remains a challenging problem.

An alternative to the shape based object classification is visual object recognition [28–32]. Visual object recognition relies on the visual appearance of an object for recognition and, works directly with the image representation of an object. In these approaches, invariant spatial features are first extracted from the image of the object. A similarity measure based on example objects or a classifier is then used to recognize the object. The most popular approach to visual object recognition is the visual bag-of-words. In the visual bag-of-words approach, each image is represented as a histogram of quantized image features. The approach discards all the spatial information about the features, which simplifies the recognition framework. However, the approach gains invariance to object view point by doing this.

1.4 Problems and Organization

This dissertation makes contributions to the above areas, which are key to scene interpretation and understanding. The specific problems addressed are:

1. Segmenting a dynamic scene captured by a stereo camera by detecting an unknown number of planar surfaces from stereo disparity.
2. Interpreting a dynamic scene captured by a moving camera by segmenting an unknown number of objects from sparse image correspondences.
3. Improving visual bag-of-words approach by reintroducing spatial information the while maintaining the simplicity of the approach.

This dissertation is organized as follows: Chapter 2 presents multi-stage merging for stereo disparity segmentation. Chapter 3 discusses structure-and-motion segmentation with multi-hypotheses clustering. Chapter 4 analyzes the computational complexity for the branch-and-bound clustering algorithm proposed in chapter 3. Chapter 5 presents the proposed two-step visual object recognition scheme. The dissertation concludes in chapter 6 with suggestions for future work.

CHAPTER 2

STEREO DISPARITY SEGMENTATION

2.1 Introduction

The ubiquitous nature of planar surfaces in indoor as well as outdoor environments has driven the research related to their detection and segmentation. For example, indoor images contain walls, floors, ceilings, etc.; outdoor images can contain the ground, sidewalks, roads, walls, roofs, etc. Additionally, objects, such as humans and vehicles, can appear to be planar if they are observed from a distance. The presence of planar surfaces can ease various tasks such as mobile navigation [33, 34], scene structure segmentation [35], and camera self-calibration [36]. To mention a few approaches, the planar surface detection can be carried out with sparse image features [37, 38], optical flow vectors [39], 3D range data [10, 40], or stereo disparity [8, 9].

The disparity based planar surface segmentation problem can be formulated as a clustering problem similar to [9] or [10], where the number of clusters is known in advance. However, this formulation of little use to interpret scenes, for which the number of objects is unknown. An interesting alternative formulation for clustering is provided by Veeman *et al.* [41, 42] known as maximum variance clustering that automatically chooses the number of clusters. According to [42], objective clustering requires scale information, which can be provided by maximum allowable cluster variance. Region growing and split-and-merge algorithms also rely on the maximum variance criterion. Our method adapts the maximum variance criterion in a split-and-merge clustering framework and selects the number of planar surfaces au-

tomatically as a byproduct of the merging process. The split-and-merge or region growing algorithms generally start from seed regions and result in a different segmentation if the seed regions are changed. To remove the suboptimality in the merging, we apply a branch-and-bound algorithm.

A branch-and-bound approach [43] to the global optimization splits the optimization problem into smaller subproblems. For these subproblems, upper and/or lower bounds of a cost function are estimated. These bounds are used to eliminate the subproblems that would not lead to an optimal solution. The subproblems that survive are further divided and the bound calculation is continued until all the subproblems are explored. The branch-and-bound procedures are applied in diverse areas such as optimal feature subset selection [43–45], image registration [46], rate-distortion based coding [47], job scheduling [48, 49] and clustering [43, 44, 50]. However, similar to other clustering formulations, the conventional branch-and-bound formulation needs the number of clusters to be known.

In this chapter, a multi-stage branch-and-bound procedure for a variable number of clusters is proposed as a part of an iterative split-and-merge algorithm for the planar surface segmentation from the disparity. An overview of the proposed approach is as follows: an initial labeling of the disparity image is carried with conventional k-means clustering. The initial labeling is then split to form spatially continuous regions. These regions are merged using the multi-stage branch-and-bound merging. The proposed algorithm extracts a single planar surface at a time by maximizing the area of the surface under the constraint that the variance of each merged area has to be less than a fixed value. The merging process is repeated until all the regions are merged to form the planar surfaces. The number of merging stages gives the number of planar surfaces present in the image. With the updated number of planar surfaces and parameters, the split-and-merge is repeated until convergence

or, until visually acceptable segmentation is attained. Our preliminary work based on this idea appeared in [51]. The work presented in this chapter was published in [52].

The rest of the chapter is organized as follows: section 2.2 states the problem to be solved. The split-and-merge paradigm utilized is illustrated in section 2.3. The proposed branch-and-bound algorithm is formulated in section 2.4. Section 2.5 presents the experimental results for a variety of stereo images.

2.2 Stereo Disparity Segmentation Problem

When an object is captured from two different cameras, similar to the motion parallax effect, it appears to have shifted in the two views. This shift is known as the stereo disparity. The disparity is inversely proportional to the distance of the object from the camera. A simplified geometric model of the stereo imaging process is shown in figure 2.1.

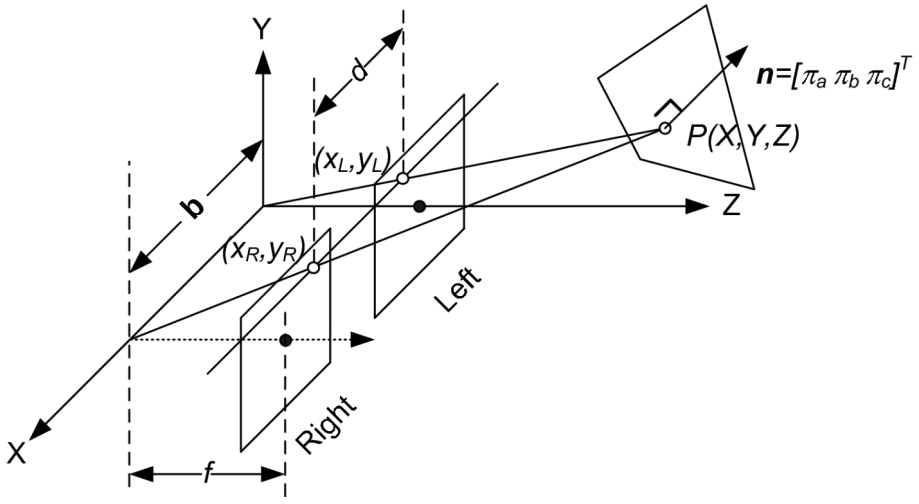


Figure 2.1. Geometry of a plane observed through a stereo camera.

An object point $P(X, Y, Z)$, which lies on a plane in a scene of interest, with normal $\mathbf{n} = [\pi_a \ \pi_b \ \pi_c]^T$ is being observed by a stereo camera. The point P is in the plane decided by the following equation [15]

$$\pi_a X + \pi_b Y + \pi_c Z + \pi_d = 0. \tag{2.1}$$

Here π_d is a constant such that $\pi_d/\|n\|$ is the distance of the plane from the origin. Image of this object point is formed at the image pixel location (x_L, y_L) in the left camera and (x_R, y_R) in the right camera. Exploiting similarity of the triangles in the stereo camera model,

$$x_L = \frac{f \cdot X}{Z}, \ y_L = \frac{f \cdot Y}{Z}, \ d = |x_L - x_R| = \frac{f \cdot b}{Z}, \tag{2.2}$$

where d is the disparity, b is the distance between the two cameras also known as the stereo baseline and f is the focal length of the camera. After inserting (2.2) in (2.1) and redefining the constants, we have:

$$d = ax + by + c, \tag{2.3}$$

where (a, b, c) are plane parameters in the disparity space. Note that the subscripts representing a camera are dropped for simplicity. Thus, the planar surfaces can be segmented by carrying out clustering in the (a, b, c) space. Before formulating the clustering algorithm in the next section, the problem is defined in the rest of this section.

Let $\mathcal{S}$ denote the pixel lattice of size M, where the disparity image $D = [d_1\ d_2\ \ldots\ d_M]^T$ is being observed. There are N planes in the image $\{P_1, P_2, \ldots, P_N\}$, such that $\mathcal{S} = \bigcup_{k=1}^{N} P_k$. The corresponding label field is given by

$$F = [f_1\ f_2\ \ldots\ f_M]^T,\ f_i \in \mathcal{L}, \tag{2.4}$$

where $\mathcal{L} = \{1, 2, \ldots, N\}$ is a set of possible plane labels. The disparity at a pixel $i \in \mathcal{S}$ is thus given by:

$$d_i = \sum_{k=1}^{N} \{a_k x_i + b_k y_i + c_k\} h_{ki} + n_i. \tag{2.5}$$

Here, (x_i, y_i) is the spatial location of the pixel i. h_{ki} is an indicator function such that $h_{ki} = 1 \Leftrightarrow f_i = k$. The parameters $\theta_k = [a_k\ b_k\ c_k]^T$ are the coefficients of the equation of the plane P_k in the disparity space. The observation is assumed to be corrupted by Gaussian noise n_i with zero mean and standard deviation σ_i to account for inaccuracies in estimation of disparity. Since the uncertainty in the disparity estimation can arise from the lack of texture, a measure that judges the "texturedness" of the local image can be used to initialize σ_i. Such measures include good features to track [53] and Harris corner detector [54]. To simplify the model, σ_i is replaced by σ for all i making the noise independent and identically distributed (iid). Our goal is to find the number of planes N, the corresponding labeling F and the parameters $\Theta = [\theta_1\ \theta_2\ \ldots\ \theta_N]$, given the disparity image D.

2.3 Segmentation Methodology

This section elaborates on the proposed method, which follows a split-and-merge paradigm. The splitting is accomplished with spatial continuity and the merg-

ing is carried out under a constraint of maximum allowable variance for a cluster. The following two subsections explain the split-and-merge steps of the algorithm.

2.3.1 Split

In the split step, it is assumed that the number of planar surfaces N and the corresponding plane parameters Θ are known either due to their initialization or as a result of the multi-stage merge in the previous iteration. Given the parameters Θ, the label field F can be found as,

$$\begin{aligned} \hat{F} &= \arg\max_{F} \Pr(F|\Theta, D), \\ &= \arg\max_{F} \Pr(D|F)\Pr(F). \end{aligned} \tag{2.6}$$

With a noninformative prior for the label field, (2.6) reduces to a maximum likelihood estimate. The estimate for the label of each pixel can be computed as:

$$\hat{f}_i = \arg\min_{f} \left\{ [d_i - (a_f x_i + b_f y_i + c_f)]^2 \right\}. \tag{2.7}$$

An estimate for the label field can thus be generated by minimizing the residuals between the actual disparity and the modeled disparity. The planar surfaces generated after the labeling are expected to be spatially continuous. As the optimization in (2.7) does not enforce spatial continuity, the label field generated is not necessarily spatially continuous. The label field F is split into N_s regions based on the spatial continuity criterion with connected component analysis.

The least square estimate for the plane parameters $\theta_k = [a_k \ b_k \ c_k]^T$ for the region P_k can be calculated as,

$$\begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix} = \begin{bmatrix} \phi_k^1 & \phi_k^4 & \phi_k^6 \\ \phi_k^4 & \phi_k^2 & \phi_k^5 \\ \phi_k^6 & \phi_k^5 & \phi_k^3 \end{bmatrix}^{-1} \begin{bmatrix} \phi_k^7 \\ \phi_k^8 \\ \phi_k^9 \end{bmatrix}. \tag{2.8}$$

Here,

$$\begin{aligned} &\phi_k^1 = \sum_{i \in P_k} x_i^2, \phi_k^2 = \sum_{i \in P_k} y_i^2, \phi_k^3 = \sum_{i \in P_k} 1 = |P_k| \\ &\phi_k^4 = \sum_{i \in P_k} x_i y_i, \phi_k^5 = \sum_{i \in P_k} y_i, \phi_k^6 = \sum_{i \in P_k} x_i, \\ &\phi_k^7 = \sum_{i \in P_k} x_i d_i, \phi_k^8 = \sum_{i \in P_k} y_i d_i, \phi_k^9 = \sum_{i \in P_k} d_i, \\ &\phi_k^{10} = \sum_{i \in P_k} d_i.^2 \end{aligned}$$

The corresponding sum of residuals for the region P_k is given by,

$$\begin{aligned} r_k^2 = [&\phi_k^{10} + a_k^2 \phi_k^1 + b_k^2 \phi_k^2 + c_k^2 \phi_k^3 + 2a_k b_k \phi_k^4 + \\ &2b_k c_k \phi_k^5 + 2a_k c_k \phi_k^6 - 2a_k \phi_k^7 - 2b_k \phi_k^8 - 2c_k \phi_k^9]. \end{aligned} \tag{2.9}$$

In the merge phase, only the variables Φ_k and the sum of residuals r_k^2 defined here need to be dealt with, rather than the individual pixel values. This significantly reduces the computational burden in the merge phase.

2.3.2 Merge

In this section, the main contribution of this chapter, a multi-stage merging strategy, is proposed that can automatically detect the number of planar surfaces. At

each stage, a planar surface is extracted by merging regions P_k under the constraint that the variance of the planar surface formed by merging η^2 is less than the maximum allowable variance B. According to the notion of maximum variance clustering, the value of B is used to select the scale of segmentation and has to be provided by the user. Since the variance of a cluster is the result of the variations in the signal and the noise involved in the acquisition, the knowledge of both must be applied to select an appropriate value of B. The average residuals η^2, for the surface extracted at the j^{th} stage $\mathcal{P}_j$, can be calculated as,

$$\eta^2(\mathcal{P}_j) = \frac{1}{|\mathcal{P}_j|} \sum_{i \in \mathcal{P}_j} ||d_i - (a_j x_i + b_j y_i + c_j)||^2 \leq B, \tag{2.10}$$

where (a_j, b_j, c_j) can be computed from the ϕ_k values of the merged regions as,

$$\begin{bmatrix} a_j \\ b_j \\ c_j \end{bmatrix} = \begin{bmatrix} \sum \phi_k^1 & \sum \phi_k^4 & \sum \phi_k^6 \\ \sum \phi_k^4 & \sum \phi_k^2 & \sum \phi_k^5 \\ \sum \phi_k^6 & \sum \phi_k^5 & \sum \phi_k^3 \end{bmatrix}^{-1} \begin{bmatrix} \sum \phi_k^7 \\ \sum \phi_k^8 \\ \sum \phi_k^9 \end{bmatrix}. \tag{2.11}$$

The summations in the above matrix equation are carried over $P_k \in \mathcal{P}_j$. However, this constraint alone does not yield a unique solution. While obeying this constraint, the area of the extracted surface is maximized. The area criterion can be expressed in terms of the variable ϕ_k values as,

$$A(\mathcal{P}_j) = \sum_{P_k \in \mathcal{P}_j} \phi_k^3. \tag{2.12}$$

Thus, at each stage j we want to choose an optimal merged surface $\mathcal{P}_j$, which is a subset of $P = \{P_1, P_2, \ldots, P_{N_s}\}$, such that its area $A(\mathcal{P}_j)$ is maximized and the variance $\eta^2(\mathcal{P}_j)$ remains under a fixed value B. Once such a subset is determined,

this optimal subset is extracted as $\mathcal{P}_j^*$ and its members are removed from the set P to update P as $P = P \setminus \mathcal{P}_j^*$. N_s is also updated to $N_s = N_s - |\mathcal{P}_j^*|$. The merging process is repeated for the updated values of P and N_s until P is empty.

After a successful merging phase, an updated number of planar surfaces N is available, which is the same as the number of merging stages in the iteration. The corresponding planar surface parameters can be calculated from (2.11) where the summations are carried over $P_k \in \mathcal{P}_j^*,\ j = 1, 2, \ldots, N$. If the labeling results do not converge, then the split-and-merge procedure can be repeated using the updated value of N and the planar surface parameters.

In the next section, a branch-and-bound algorithm is formulated to merge regions optimally and efficiently.

2.4 Multi-Stage Branch-and-Bound Merging

Let $\mathcal{P}_j = \{z_1, z_2, \ldots, z_n\}$ denote the set of n elements, which optimizes the merging criterion at the j^{th} stage. For convenience, hereon the regions are indicated with their indices alone, i.e., $z_1, z_2, \ldots, z_n \in \{1, 2, \ldots, N_s\}$. The number of possible solutions for each stage of the merging is given by,

$$\sum_{i=0}^{N_s} \binom{N_s}{i} = 2^{N_s}, \tag{2.13}$$

which increases exponentially with N_s. This makes it difficult to evaluate all the solutions even for a small number such as 20 or 30. This optimization can be carried out efficiently with a branch-and-bound procedure [43].

The solutions for the problem can be represented as a rooted tree. Each node of the tree gives one possible solution for the problem. The regions included in the solution can be found by the walk from the root of the tree to the current node. If the

root of the tree represents an empty set, then each node encountered is added to the set of its parent node to generate the solution. Alternatively, the encountered nodes are removed if the root node includes all the possible regions. In our formulation, we choose an empty set as the root node. For the merging problem, the probability of encountering a solution is higher near the root when this formulation is used. This greatly speeds up the search for the optimal solution.

It is important that every solution is listed only once to avoid unnecessary computations. This can be ensured by creating child nodes that are different than:

- left siblings,
- ancestors,
- left siblings of ancestors.

One simple way of generating such a solution tree for $N_s = 4$ is shown in figure 2.2. Note that, in the solution tree for $N_s = n$, $z_1 < z_2 < z_3 \ldots < z_n$ and (left sibling $<$ right sibling). These two conditions ensure that the rule stated above to generate the child nodes is followed. The current solution $\{z_1, z_2, z_3 \ldots, z_n\}$ is a subproblem for all its descendants. Additionally, all the solutions representing the predecessors of the current solution are subproblems for $\{z_1, z_2, z_3 \ldots, z_n\}$.

Before the branch-and-bound algorithm is formulated, we estimate the bounds on the best solution that can be reached from the current node. Consider a solution at a node to be $\mathcal{P}_j = \{z_1, z_2, \ldots, z_n\}$. The best solution in terms of area, i.e. the maximum area that the node can lead to, is given by the sum of the areas of the regions at the current node and the areas of all the regions with index greater than z_n.

$$A_{\max}(\mathcal{P}_j) = A(\mathcal{P}_j) + A(z_n + 1, z_n + 2, \ldots, N_s). \tag{2.14}$$

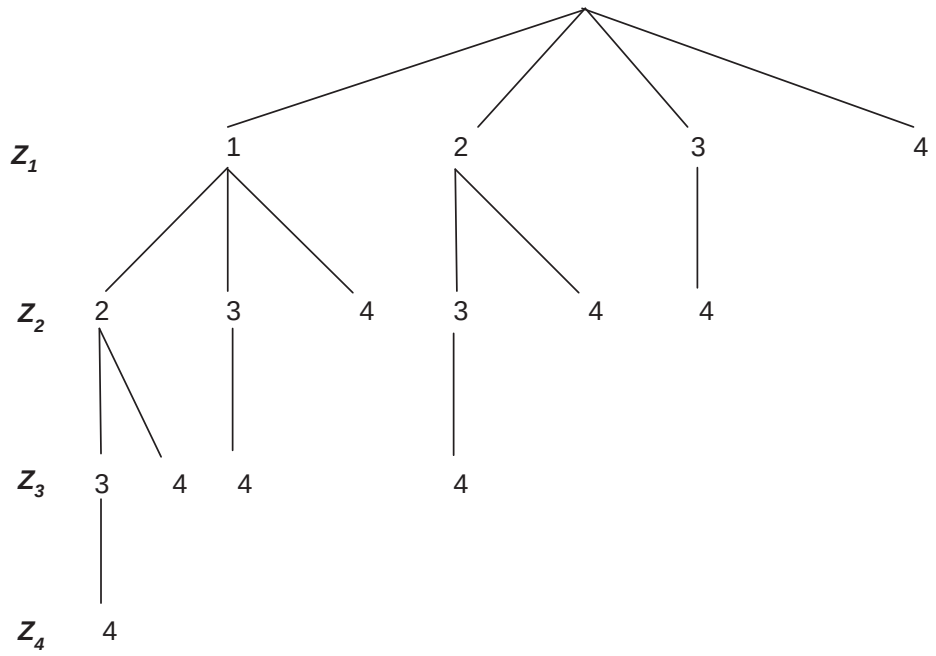


Figure 2.2. Solution tree for N_s=4.

Clearly, if $A_{\max}$ is smaller than the present optimal value for area A_j^*, then the current node cannot lead to a better solution and the child nodes of this node can be safely excluded from the search.

On the other hand, the best solution in terms of the variance minimizes the average residuals. The variance does not increase or decrease monotonically with the tree depth, therefore a bound on its value cannot be derived directly. However, the sum of residuals (before normalization by the area) increases monotonically with the depth and can be used to construct a bound on the variance. The bound on the variance is given by the ratio of the lower bound on value of residuals of the child nodes and upper bound on area of the child nodes, i.e. $A_{\max}$.

$$\eta^2_{\min}\left(\mathcal{P}_j\right) = \frac{|\mathcal{P}_j|\eta^2(\mathcal{P}_j) + \min\{r^2_{z_n+1}, r^2_{z_n+2}, \ldots, r^2_{N_s}\}}{A_{\max}}. \tag{2.15}$$

If $\eta^2_{\min}$ is greater than the bound B, then the current parent node will not lead to the optimal solution and search for the optimal solution can be terminated at this node.

An additional bound can be derived when at least one stage of merging has finished, i.e. when $j > 1$. As $\mathcal{P}^*_j$ is extracted by maximizing the area under the same bound B as the one used for $j-1$, the area of the optimal solution in any stage A^*_j must be less than or equal to the optimal area achieved in the previous stage A^*_{j-1}. Thus, an upper bound on A^*_j, i.e. A_{upper}, can be imposed as,

$$A^*_j \leq A_{\text{upper}} = A^*_{j-1}. \tag{2.16}$$

A multi-stage branch-and-bound merging algorithm based on the above bounds is listed below.

1. Overall initialization: Set stage $j = 1$, the number of planar surfaces $N = 1$, and the upper bound on the area $A_{\text{upper}} = \infty$.
2. Stage initialization: Set number of spatially continuous regions $N_s = |P|$, the optimal area A^*_j=0, the optimal merging $\mathcal{P}^*_j = \emptyset$, the tree level $i = 1$ and the current node $z_0 = 0$.
3. Generate Successors: Initialize $LIST(i)$,

$$LIST(i) = \{z_{i-1} + 1, z_{i-1} + 2, \ldots, N_s\}. \tag{2.17}$$

4. Select new node: If $LIST(i)$ is empty, go to step 6. Otherwise, set $z_i = k$ where $k \in LIST(i)$. Set the current solution $\mathcal{P}_j = \{z_1, z_2, \ldots, z_i\}$. Delete k from $LIST(i)$.

5. Check bounds: Compute $A_{\max}(\mathcal{P}_j)$. If $A_{\max}(\mathcal{P}_j) < A_j^*$, go to step 6. Calculate $\eta^2_{\min}(\mathcal{P}_j)$. If $\eta^2_{\min}(\mathcal{P}_j) > B$, go to step 6. Compute $A(\mathcal{P}_j)$. If $A_{\text{upper}} < A(\mathcal{P}_j)$, go to step 6. Compute $\eta^2(\mathcal{P}_j)$. If $A^* \leq A(\mathcal{P}_j)$ and $\eta^2(\mathcal{P}_j) \leq B$, set $A^* = A(\mathcal{P}_j)$ and $\mathcal{P}_j^* = \mathcal{P}_j$. Set $i = i + 1$ and go to step 3.
6. Backtrack to the lower level: Set $i = i - 1$. If $i > 0$, go to step 4. If $A_j^* = 0$, set $\mathcal{P}_j^* = \{1\}$. Set $A_{\text{upper}} = A_j^*$. Update $P = P \setminus \mathcal{P}_j^*$. If $P = \emptyset$, terminate the algorithm. Set $j = j + 1$, $N = N + 1$ and go to step 2.

Figure 2.3 gives the flowchart for the multi-stage branch-and-bound merging process for a better understanding.

2.5 Experimental Results

The proposed planar surface segmentation process was tested with a variety of synthetic and real data. Before proceeding to the results, we briefly describe the initialization strategy for the algorithm. During the initialization, we utilize the fact that the planar objects are expected to have constant or piecewise continuous values of disparity. To initialize the planar surface parameters, the cumulative histogram of the disparity image was split into N_{init} equal segments. First, the disparity values for the image are arranged in ascending or descending order. The disparity values are then separated in N_{init} number of bins each carrying an equal number of pixels, i.e. the first $\frac{M}{N_{\text{init}}}$ pixels in the ordered pixels are assigned to plane 1, next $\frac{M}{N_{\text{init}}}$ are assigned to plane 2 and so on.

Unlike the conventional iterative methods (e.g. expectation-maximization, generalized k-means, and segmentation-estimation), which carry out the labeling and the segmentation in separate steps, our method carries out these steps simultaneously. Iterations for the method are required to capture the boundaries that are not cap-

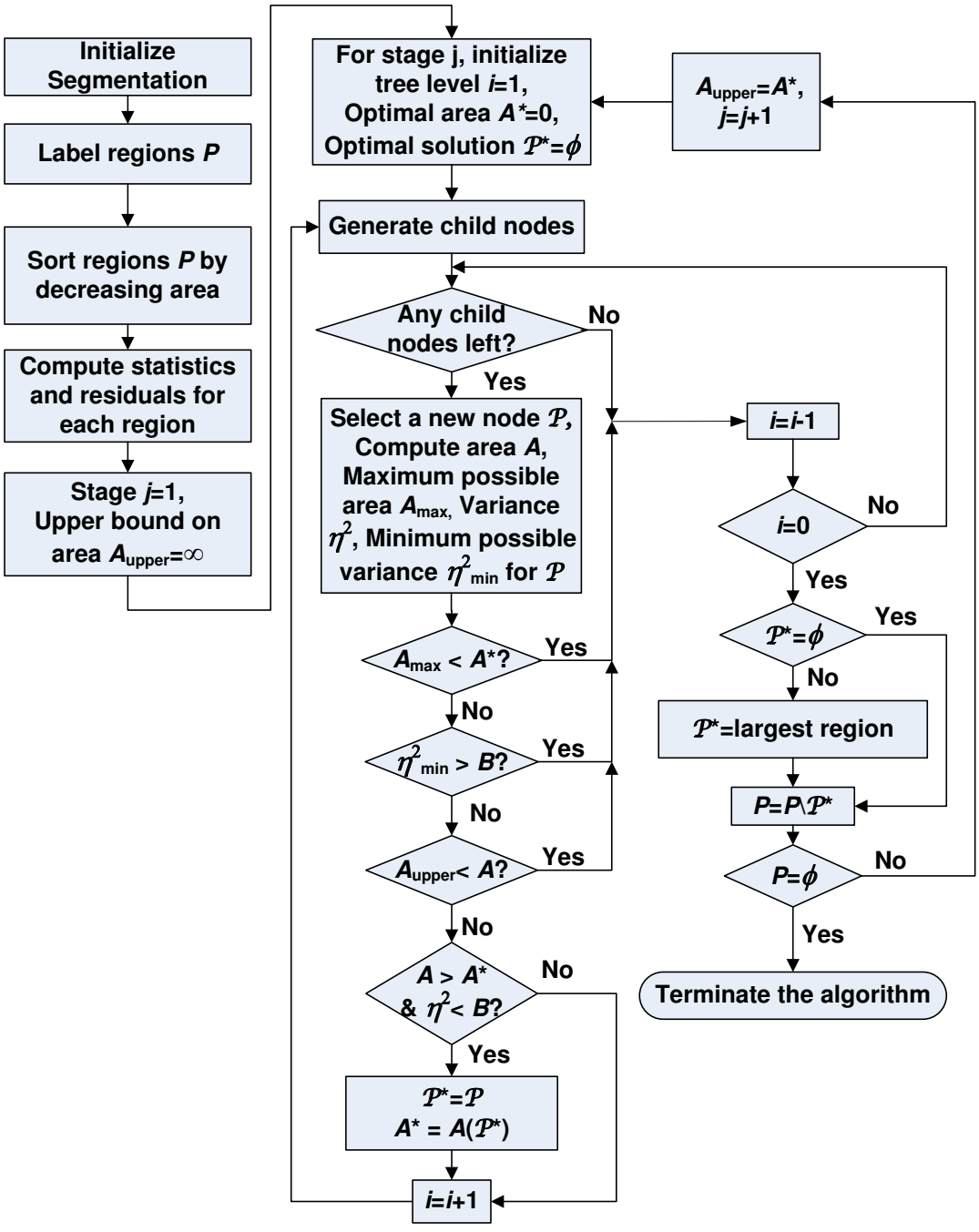


Figure 2.3. Flowchart for multi-stage branch-and-bound merging.

tured by the initialization. Thus, a very small number of iterations (typically 4) is required to produce visually acceptable segmentation.

2.5.1 Middlebury College Stereo Images

The first set of experiments was carried out with the stereo image pairs and their ground truth disparity available at the Middlebury stereo vision research page [55]. For each image, the number of planar surfaces is known and this information can be used to verify the success of the proposed method.

Figures 2.4(a) and (b) show the left and right images of the "Barn" sequence. Each image in this sequence is of size 432×381. The image shows 6 planar surfaces of various size, shape and orientation. First, the ground truth disparity (shown in figure 2.4(c)) was segmented with bound $B = 0.1$. The number of surfaces was initialized to be $N_{\text{init}} = 10$. For each segment, the planar parameters were estimated, which were used for the spatial continuity based splitting in the first iteration of the algorithm. The planar surfaces extracted after 4 iterations of the split-and-merge are shown in figure 2.4(d).

The planar surface extraction was repeated for the estimated disparity map. To calculate the disparity map, the sum of absolute differences (SAD) was minimized over a window shifted along the epipolar lines of the image. Additionally, to remove the outliers, including the occluded areas, we employ the left-right check [56]. For the "Barn" images, a window of size 7×7 was used for the matching. Finally, a subpixel value of the disparity was estimated by fitting a parabola to the SAD values around the integer disparity [57].

Figure 2.4(e) shows the estimated disparity for the sequence. The small dark areas around the edges of the planes are the occluded areas, for which a reliable estimate of the disparity is unavailable. The detected planar surfaces are shown in

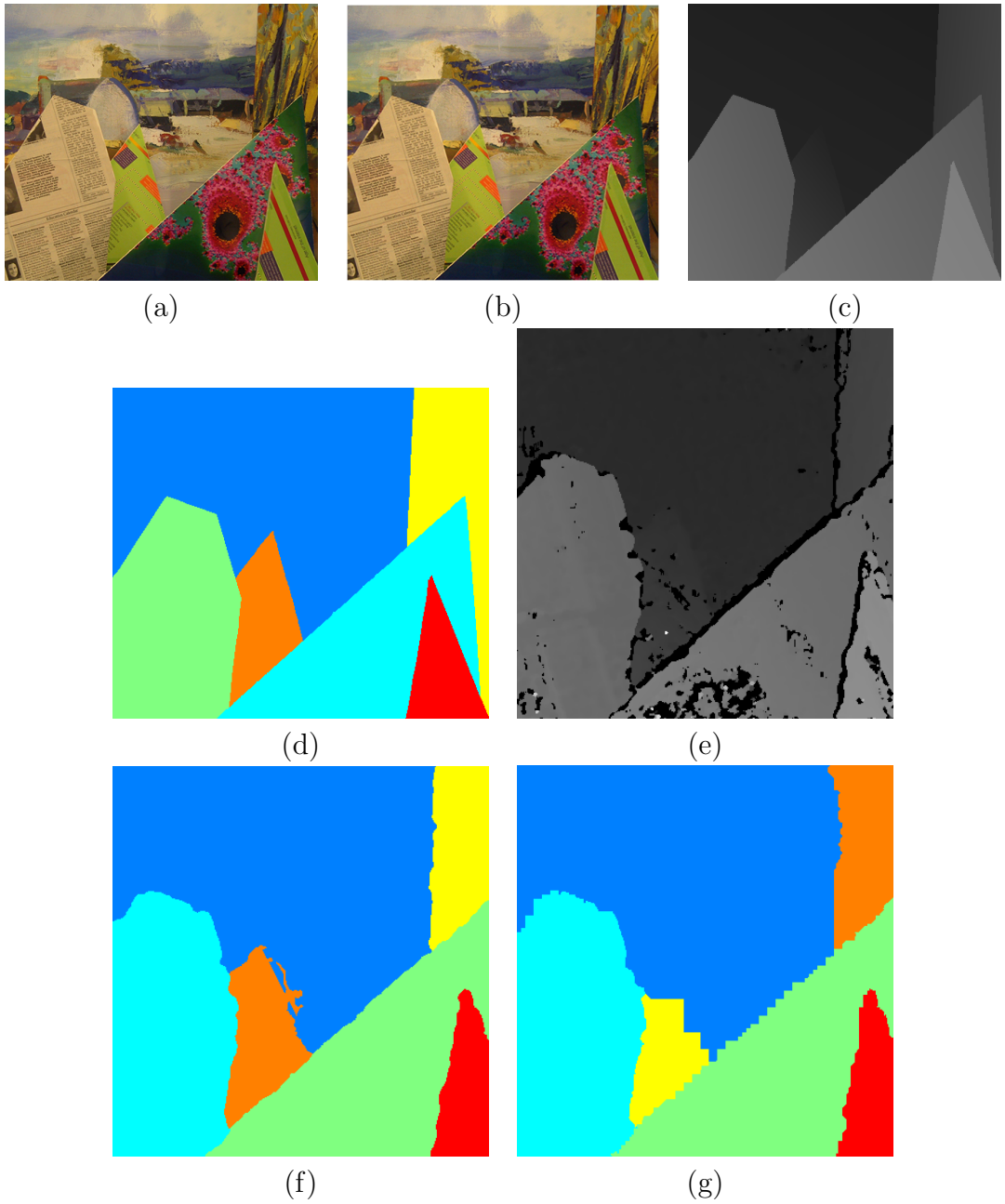


Figure 2.4. (a) Left image of pair "Barn," (b) Right image of pair "Barn," (c) Ground truth disparity, (d) Detected planar surfaces ($B = 0.1$, iterations=4), (e) Calculated subpixel disparity, (f) Detected planar surfaces ($B = 0.05$, iterations=4, Vinet's measure=0.0396), (g) Detected planar surfaces with split-and-merge ($B = 0.05$, Vinet's measure= 0.0533).

figure 2.4(f) after some postprocessing. The postprocessing involved size filtering on the detected regions. Any regions of size less than 50 pixels were filtered out. These regions along with the occluded regions were then labeled by nearest neighbor interpolation. The foreground planes are bigger compared with the ones in the segmented ground truth. This is due to the use of the square window to calculate the disparities. Additionally, note that the disparities at extremities of the image, especially the left and right edges, cannot be estimated. For this reason, the calculated disparity images are smaller than the ground truth disparity images.

For comparison, a quad-tree based split-and-merge algorithm was implemented [58]. The steps involved in the algorithm are listed below:

- Split the region into four quadrants if the variance is greater than B.
- If no further splitting is possible then combine the neighboring regions such that the average residuals of the combined regions are less than B.
- Stop if no regions can be merged.

Note that apart from the splitting and merging strategy used, this algorithm follows the same computational framework as our algorithm. Figure 2.4(g) shows the result of the planar surface segmentation with quad-tree split-and-merge segmentation. Vinet's measure [59, 60] is used for the objective comparison between both the methods. For the "Barn" images, the proposed method slightly outperforms (by $\sim$1%) the quad-tree split-and-merge method according to Vinet's measure. Additionally, the planar surface boundaries have a jagged appearance in the quad-tree based method due to the splitting method used.

Table 2.1. Solutions explored in the first stage for the ground truth "Barn" disparity

Iteration	N	N_s	Solutions explored	Fraction explored
1	7	37	1987	1.44e-8
2	6	30	1032	9.61e-7
3	6	22	237	5.65e-5
4	6	23	269	3.21e-5

Table 2.2. Solutions explored in the first stage for the calculated subpixel "Barn" disparity

Iteration	N	N_s	Solutions explored	Fraction explored
1	7	40	1462	1.33e-9
2	6	40	4092	3.72e-9
3	6	23	224	2.67e-5
4	6	21	146	6.96e-5

Tables 2.1 and 2.2 show the number of solutions explored to extract the largest planar surface from the ground truth and the calculated disparity. The fraction explored is calculated as,

$$\text{Fraction explored} = \frac{\text{Solutions explored}}{2^{N_s}}. \tag{2.18}$$

The tables demonstrate the effectiveness of the proposed branch-and-bound algorithm that explores a very small fraction of the solution space to reach the optimal solution. Before the multi-stage merging was carried out, all the spatially continuous regions were sorted according to decreasing area. To speed up the solution, regions smaller than 50 pixels were removed from the search. If the number of regions left in the search after this was greater than 40, then the 40 largest areas were considered for the merging operation to speed up the process further. Note that even with 40

areas, the number of possible solutions is $2^{40} \approx 1.099e12$. The tables also show that the number of planar surfaces is correctly identified as 6 in both the cases.

Figures 2.5 and 2.6 show additional results with the "Poster" and "Venus" images, both with the ground truth and the calculated disparities. The number of planar surfaces detected in the "Poster" image was six for the ground truth as well as the computed disparity. Five planar surfaces are detected in the "Venus" ground truth disparity. However, for the smallest planar surface, which lies in the right bottom corner of the image, the disparity cannot be calculated. Thus, only four surfaces are detected in the calculated disparity. For the "Poster" and "Venus" images, the proposed method significantly outperforms the quad-tree split-and-merge method. This can be seen visually as well as through the Vinet's measure. The quad-tree split-and-merge oversegments the image even though it uses the same value of maximum allowable variance B as the proposed method. This is mainly due to the suboptimal merging the method uses.

2.5.2 JISCT Stereo Images

Images chosen in the first experiment contained only planar objects and the scenes were not natural. The second set of experiments was carried out with some of the JISCT stereo images [61], which contain a few real life sequences with non-planar objects. The first sequence is called "Shrub" and its left image is shown in figure 2.7 (a). The dimensions of the image sequence are 512×480. The images contain a hedge in front of a building wall along with a parking sign and a part of a road at the bottom.

Similar to the first experiment, N_{init} was chosen to be 10. The initial segmentation is shown in figure 2.7(c). After four iterations of the split-and-merge with bound $B = 0.5$, three planar surfaces are detected. One corresponds to the wall,

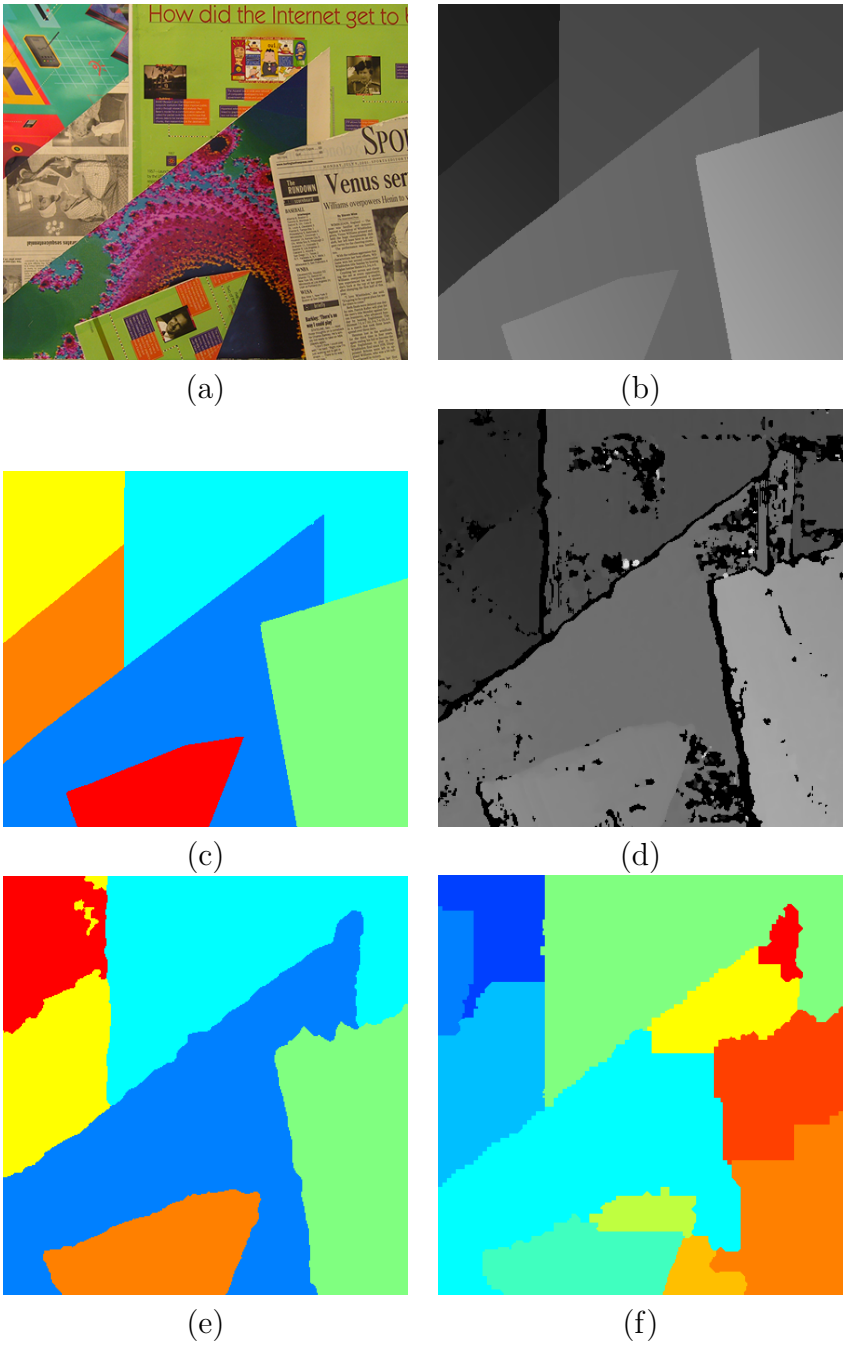


Figure 2.5. (a) Left image of pair "Poster," (b) Ground truth disparity, (c) Detected planar surfaces ($B = 0.1$, iterations=4), (d) Calculated subpixel disparity, (e) Detected planar surfaces ($B = 0.05$, iterations=4, Vinet's measure=0.0419), (f) Detected planar surfaces with split-and-merge ($B = 0.05$, Vinet's measure= 0.2448).

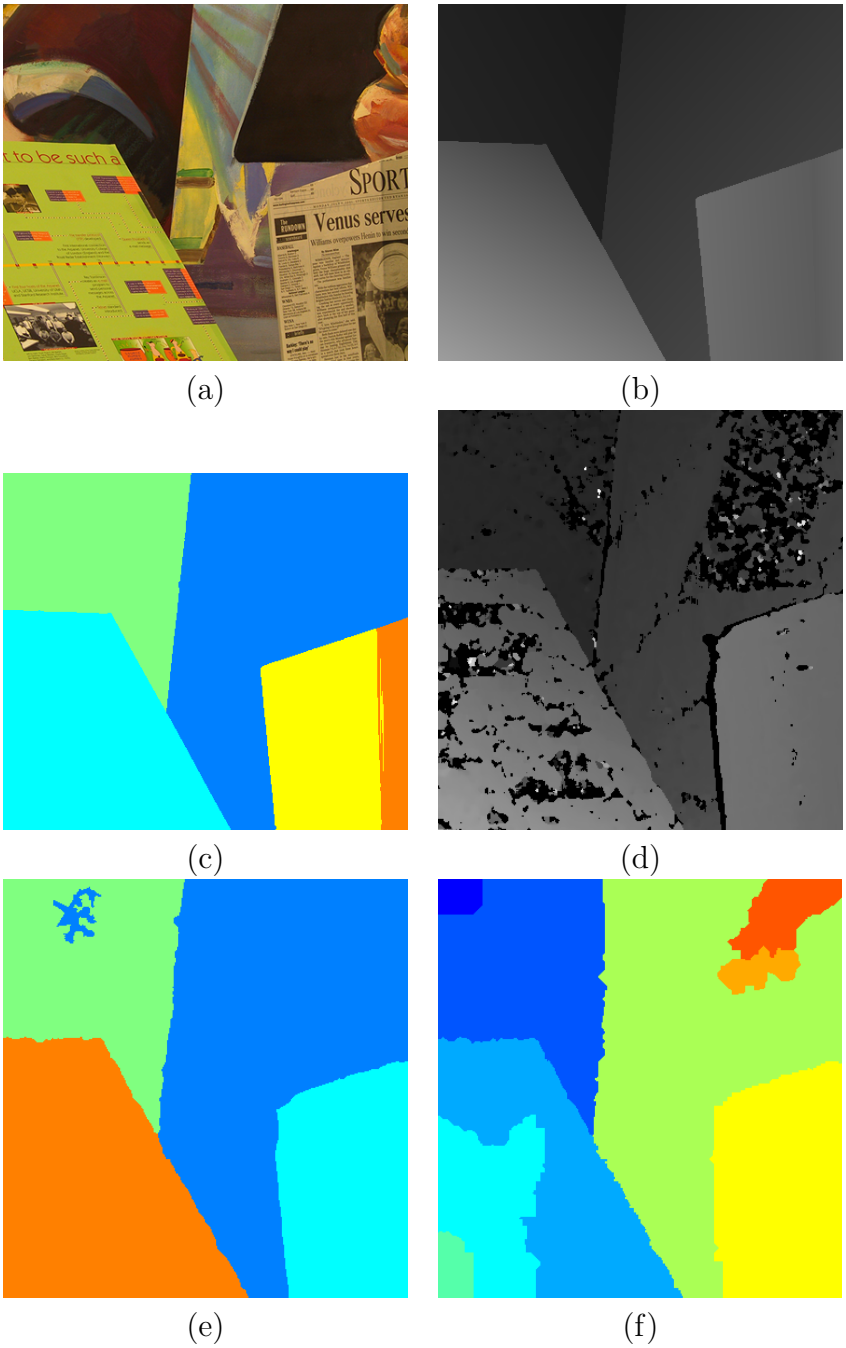


Figure 2.6. (a) Left image of pair "Venus," (b) Ground truth disparity, (c) Detected planar surfaces ($B = 0.1$, iterations=4), (d) Calculated subpixel disparity, (e) Detected planar surfaces ($B = 0.05$, iterations=4, Vinet's measure=0.0319) , (f) Detected planar surfaces with split-and-merge ($B = 0.05$, Vinet's measure= 0.1890).

the econd corresponds to the hedge, and the last represents the part of the road. Table 2.3 shows the number of solutions explored in the first stage of the algorithm for the images. Due to the increased complexity of the scene compared to the first experiment, the maximum number of solutions is explored in the first stage of the first iteration, i.e. 212644 for $N_s = 40$ (fraction 1.93e-7). This large number is due to the presence of numerous small regions and very few large regions. As more regions are needed to form the optimal solution, the optimal solution will be present farther from the root, which leads to a higher number of solutions being explored.

The other image sequence used for the experiment was "Parking meter." The dimensions of this image sequence are 512×480 as well. The left side of the image shows a hedge in front of a wall. Additionally, a few parking meters are visible with a part of a car. The plane detection results for $B = 0.3$ after four iterations are shown in figure 2.8 (d). Three separate planes are detected for the hedge, one parallel to the camera image plane, the second parallel to the wall, and the last very close to the camera. The partially visible car is also labeled to be in a plane parallel to the image plane. Two different planes for the two parallel wall surfaces can also be seen. This scene is more challenging than the others due to its complex structure. This complexity is also reflected in the number of solutions explored in Table 2.4. As seen in figure 2.8(e), the quad-tree based split-and-merge performs very badly for the scene.

2.5.3 University of Bologna Stereo Sequences

Finally, the proposed method was tested with "Indoor" and "Outdoor" video sequences from the University of Bologna [62]. These sequences have resolution of 640×480. The stereo disparity was calculated by matching windows of size 15×15. The bound on variance B was chosen to be 0.3 for these images.

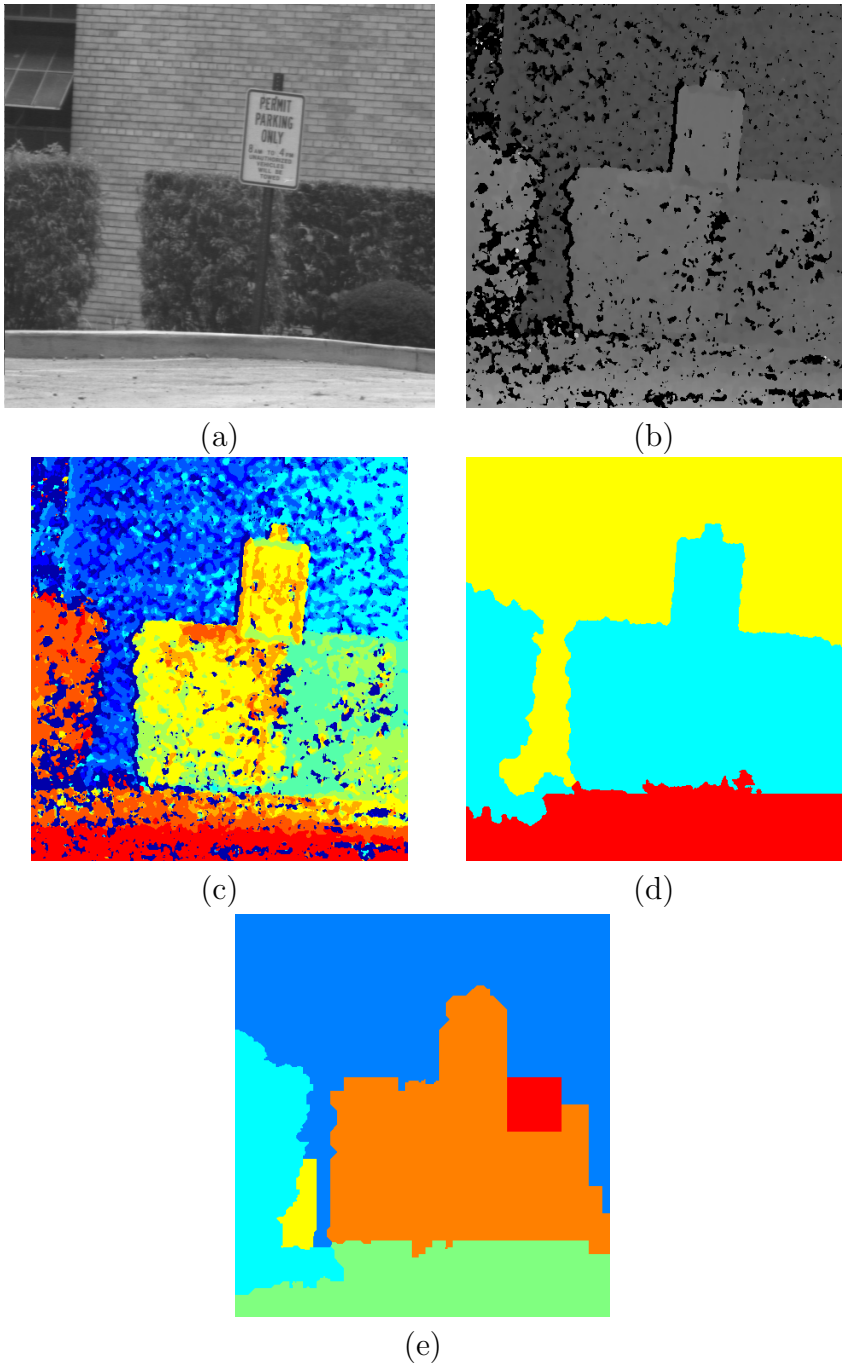


Figure 2.7. (a) Left image of pair "Shrub," (b) Calculated subpixel disparity, (c) Initial segmentation ($N_{\text{init}} = 10$), (d) Detected planar surfaces ($B = 0.5$, iterations=4), (e) Detected planar surfaces by split-and-merge ($B = 0.5$).

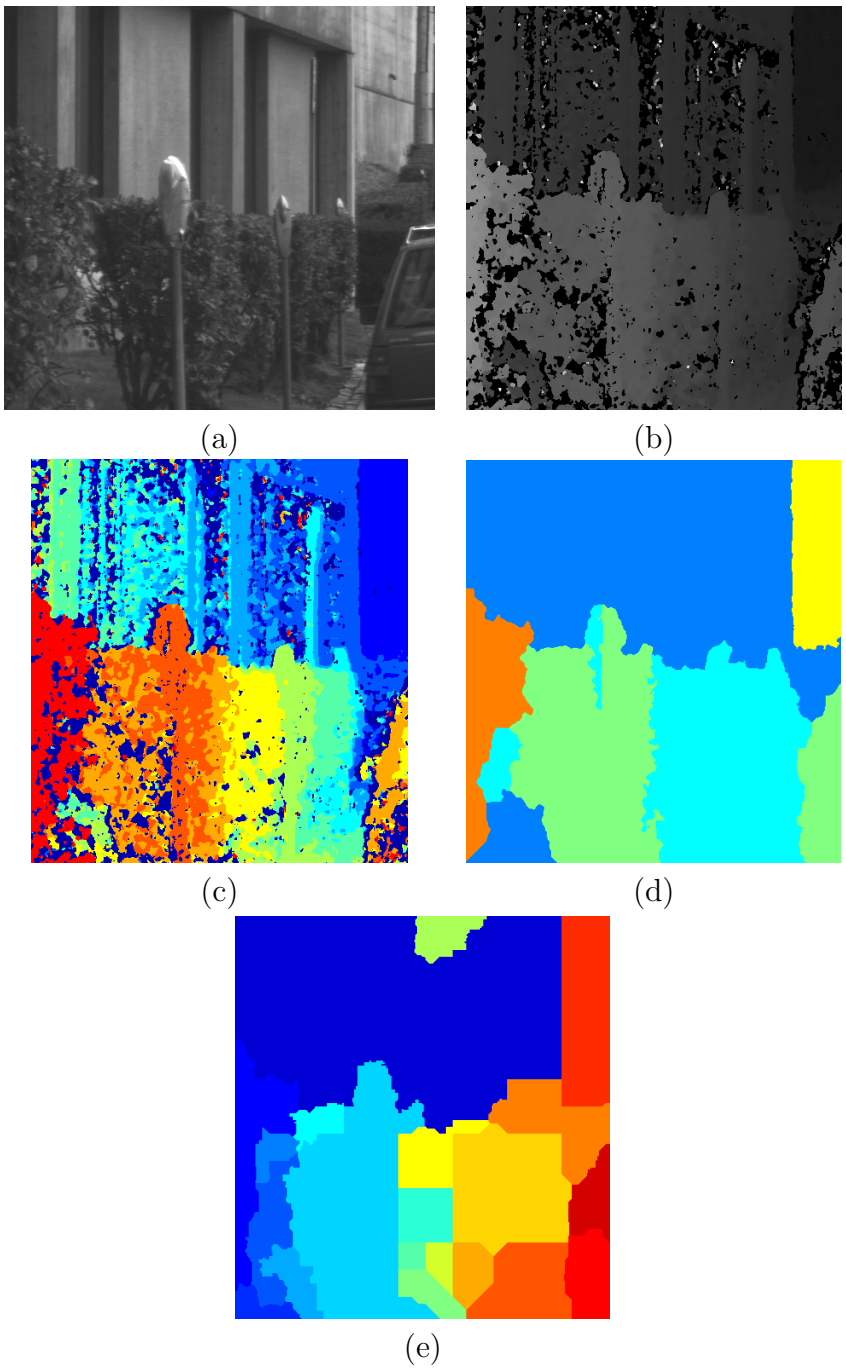


Figure 2.8. (a) Left image of pair "Parking meter," (b) Calculated subpixel disparity, (c) Initial segmentation ($N_{\text{init}} = 10$), (d) Detected planar surfaces ($B = 0.3$,iterations=4), (e) Detected planar surfaces by split-and-merge ($B = 0.3$) .

Table 2.3. Solutions explored in the first stage for "Shrub"

Iteration	N	N_s	Solutions explored	Fraction explored
1	3	40	212644	1.93e-7
2	3	40	81755	7.44e-8
3	3	34	2981	1.74e-7
4	3	34	10177	5.92e-7

Table 2.4. Solutions explored in the first stage for "Parking meter"

Iteration	N	N_s	Solutions explored	Fraction explored
1	5	40	21356	1.94e-8
2	5	40	24159	2.20e-8
3	5	40	30027	2.73e-8
4	5	40	38323	3.49e-8

Figure 2.9(a) shows frame 2 of the "Outdoor" sequence. The frame contains a staircase, a wall, and a small rectangular panel against the wall. The detected planar surfaces are shown in figure 2.9(c). Five iterations of the proposed split-and-merge procedure were applied to achieve this result.

Figure 2.9(d) shows frame 126 of the "Outdoor" sequence. In addition to the original planar surfaces, a human can also be seen in the frame. The detected planar surfaces are shown in figure 2.9(f). Six iterations of the proposed split-and-merge procedure were carried out. As a human is not a planar object, he is split into three piecewise planar patches.

The background for the "Indoor" sequence is shown in figure 2.9(g). As most of the image is textureless, the calculated disparity is sparse. However, the planes appearing in the image are correctly identified.

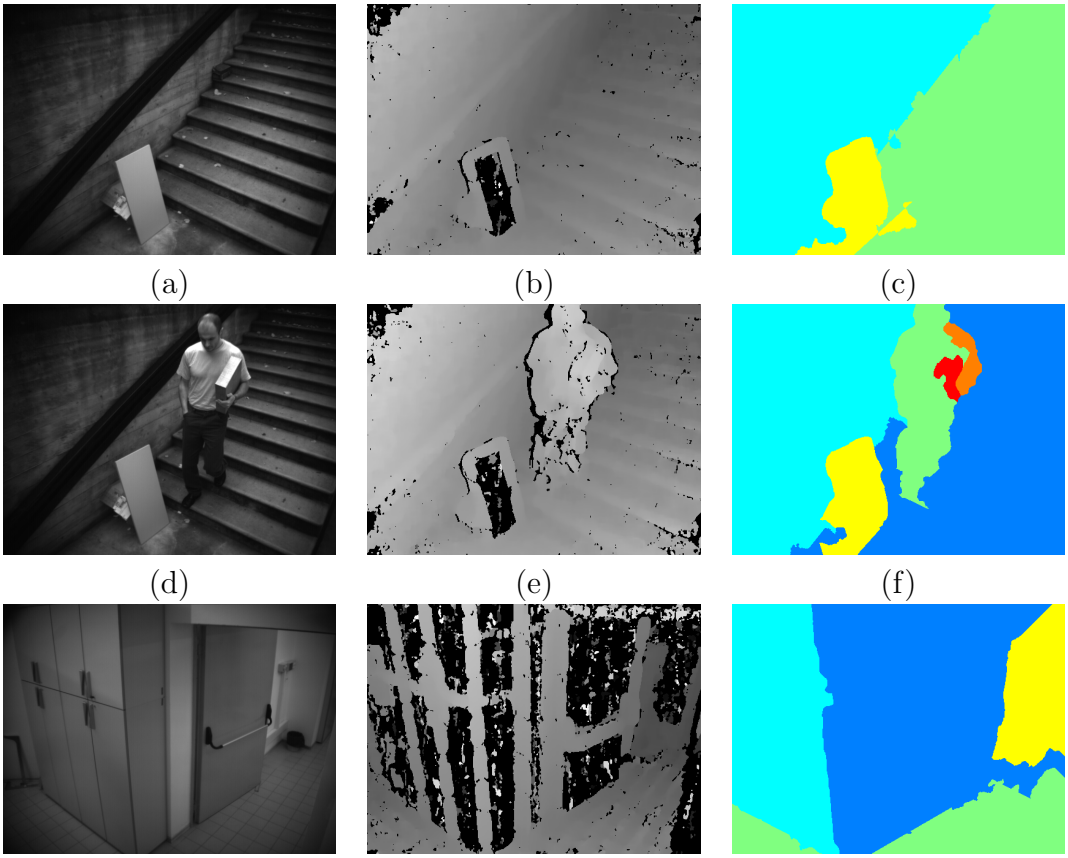


Figure 2.9. (a) Left image of pair "outdoor" frame 2, (b) Calculated subpixel disparity, (c) Detected planar surfaces ($B = 0.3$, iterations=5), (d) Left image of pair "outdoor" frame 126, (e) Calculated subpixel disparity, (f) Detected planar surfaces ($B = 0.3$, iterations=6), (g) Left image of pair "indoor" frame 10, (h) Calculated subpixel disparity, (i) Detected planar surfaces ($B = 0.3$, iterations=6) .

2.5.4 Computational Complexity

For the branch-and-bound algorithms, the worst case complexity is the same as the brute force search. However, the average case complexity, which is an indicator of performance of the algorithm, is much lesser. Table 2.5 compares the execution time for various experiments for the MATLAB implementations of the quad-tree split-and-merge and the proposed algorithm using a Dell Vostro 400 desktop with Intel Core 2 Duo 2.33 GHz (E6550) processor as a single thread. Being a global

Table 2.5. Execution time in seconds

Experiment	Quad tree	Branch and bound (4 iters.)
Barn	9.83	5.30
Poster	14.15	3.94
Venus	23.92	16.94
Shrub	17.29	55.81
Parking meter	26.85	25.12

optimization technique, execution of the proposed branch-and-bound algorithm was expected to be slower than the quad-tree split-and-merge. However, on an average the branch-and-bound algorithm outperformed the quad-tree split-and-merge.

CHAPTER 3

MULTIPLE STRUCTURE-AND-MOTION SEGMENTATION

3.1 Introduction

The disparity segmentation problem and the multiple structure-and-motion (MSaM) segmentation problem are very similar. While disparity segmentation groups pixels in a scene based on a model for their structure, MSaM segmentation groups correspondences with similar motion. A multi-stage merging formulation similar to chapter 2 is possible for structure-and-motion segmentation as well. For various reasons listed below, a new formulation is needed to carry out MSaM segmentation.

- *Nonlinear problem*: The bounds estimated in branch-and-bound merging are due to the linear nature of the structure model applied. While estimation of the bounds for the nonlinear model involved in MSaM segmentation should be possible, it is not straightforward. Possibly, these bounds would not be "tight", slowing the branch-and-bound search.
- *Density of correspondences*: The epipolar geometry between left and right views from stereo camera is fixed, while the epipolar geometry between two views captured from a moving camera changes continuously. The known epipolar geometry reduces the search area for correspondences. For calibrated and rectified stereo camera images, search along horizontal scan line is needed to establish the correspondences. For images captured from a moving camera, the correspondences have to be searched for over the entire image. Additionally, the variation of perspective between two views is larger in a moving camera compared to a stereo camera. For these reasons, matching image features is

easier for a stereo camera compared to a moving camera. Thus, a very few reliable matches are possible for a moving camera. Due to the sparsity of the correspondences, the spatial continuity based split phase of the split and merge algorithm proposed in chapter 2 is ineffective.

- *Outliers*: In addition to sparsity of matches, unknown epipolar geometry and drastic changes in perspective cause more mismatches with moving camera. Thus, an explicit treatment of these outliers becomes necessary to avoid their negative impact on MSaM segmentation.
- *Greedy formulation*: The segmentation formulation in chapter 2 is greedy and, was preferable dues to the large volume of data. As the data is sparse for MSaM segmentation, an optimal formulation for the problem is now feasible.

The MSaM segmentation problem, where the number of objects is unknown, is typically formulated as a recursive and sequential clustering problem. The clustering is carried out by varying the number of objects,i.e. clusters, and selecting an optimal number of cluster with a model selection criterion such as Bayesian information criterion (BIC) or the Akaike information criterion (AIC) [63]. At each stage, the problem is recursive as segmentation labels are required for estimation of cluster parameters and vice-versa.

This chapter formulates the MSaM problem as a one-step combinatorial optimization problem under a sampling based framework. Initially, hypotheses for structure-and-motion model are generated by local sampling of correspondences between two views. A null hypothesis is also introduced suggesting that any match can be an outlier with uniform likelihood. Next, a model selection criterion, which penalizes the likelihood of the clustering by number of clusters, is added to the framework. The model selection criterion is optimized through a branch-and-bound process to

obtain the final MSaM segmentation. Our preliminary work based on this idea was presented in [64].

The chapter is organized as follows: Section 3.2 formulates the MSaM segmentation as a combinatorial optimization problem. A branch-and-bound solution to the problem is then formulated in section 3.3. The experimental results are presented in section 3.4.

3.2 Multiple Structure-and-Motion Segmentation Problem

Consider a set of N image correspondences

$$\mathbf{X} = \{(\mathbf{x}_1, \mathbf{x}'_1), (\mathbf{x}_2, \mathbf{x}'_2), \ldots, (\mathbf{x}_N, \mathbf{x}'_N)\},$$

where $\mathbf{x}_i$ and $\mathbf{x}'_i$ are image coordinates of the ith correspondence. The relationship of various object structures and motions in the scene can be expressed as,

$$\mathbf{x}_i^{'T} \left(\sum_{j=1}^{K} \mathcal{L}_j(i) \mathtt{F}_j \right) \mathbf{x}_i = 0. \tag{3.1}$$

Here, $\mathtt{F}_j$ is the fundamental matrix [15] for jth rigid body in the scene. The indicator function $\mathcal{L}_j(i)$ is 1 when ith correspondence belongs to the jth rigid body and 0 otherwise. A label field $L = [l_1, l_2, \ldots, l_N]$ is associated with the indicator function $\mathcal{L}_j(i)$ such that, $l_i = j$ if $\mathcal{L}_j(i) = 1$. The goal of the MSaM segmentation is to estimate the label field L. Once the label field in known, the fundamental matrix $\mathtt{F}_j$ can be computed as the least square estimate as,

$$\mathtt{F}_j = \arg\min_{\mathtt{F}} \sum_{\forall i, l_i = j} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F})^2. \tag{3.2}$$

Here, d is a distance measure such as symmetric transfer error, reprojection error or Sampson approximation [15].

On the other hand, if $\mathtt{F}_j$s are known, the maximum likelihood estimate for the label of ith match is given by,

$$\hat{l}_i = \arg\min_l d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_l)^2. \tag{3.3}$$

Equations (3.2) and (3.3) represent the parameter estimation and the label estimation or segmentation steps respectively. Since these steps are interdependent, the MSaM problem can be solved iteratively to maximize the likelihood of the correspondences. Assuming that the uncertainties in the matches are normally distributed with zero mean and standard deviation σ, the log likelihood of the matches is given by,

$$\log\{\mathrm{Lik}(\mathbf{X})\} = -\frac{1}{2}\left(\frac{\mathrm{SSD}}{\sigma^2}\right) + \mathrm{Constant}, \tag{3.4}$$

where,

$$\mathrm{SSD} = \sum_{i=1}^{N} \min_{j=1}^{K} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_j), \tag{3.5}$$

This optimization procedure also assumes that the number of clusters K is known *a priori*. This assumption is unrealistic in most of the scenes. The likelihood of the correspondences is inversely proportional to the SSD given by (3.5) and it can only decrease if a new cluster is added. Since the likelihood of the correspondences increases as K is increased, the likelihood alone cannot be applied to select an optimal value of K. A model selection criterion such as the Bayesian information criterion (BIC) or the Akaike information criterion (AIC) can be utilized to select the optimal K [63]. These criteria penalize the log likelihood of the model in proportion to the

size of the model K. A generalized cost function to incorporate this idea can be written as,

$$\mathcal{C} = -2\log\{\text{Lik}(\mathbf{X})\} + \alpha \cdot K. \tag{3.6}$$

Where α is a positive constant. For BIC, $\alpha = \log(N)$ and for AIC, $\alpha = 2N$. The first term of (3.6) gives the negative log likelihood of the model, which decreases with increase in K. The second term of (3.6) is the penalty term, which increases with increase K. Thus, the minimum cost $\mathcal{C}$ compromises between likelihood and number of cluster to select optimal K.

The cost function in (3.6) can be optimized by iterative optimization of the likelihood in (3.4) for varying values of K. Finally, the optimal K can be selected to minimize the cost $\mathcal{C}$.

Alternative to this approach is a simultaneous model selection and segmentation approach. In this approach, multiple hypotheses for fundamental matrix $\mathtt{F}_j$ where $j = 1, 2, \ldots, N_h$ are generated by sampling the correspondences. Figure 3.1 shows three different motions marked with $+$, $*$ and X and the outliers marked by O. RANSAC is applied to correspondences in the circular spatial neighborhood of a correspondence to estimate $\mathtt{F}_j$. Use of the spatial neighborhood ensures that RANSAC can correctly and quickly estimate hypotheses for fundamental matrices. Once these hypotheses are known, the MSaM segmentation problem is reduced to a combinatorial optimization problem to select K hypotheses out of total N_h hypotheses. Note that there are 2^{N_h} possible solutions for this problem. Thus, even for a moderate value of N_h, an exhaustive search becomes intractable. However, the nature of the problem allows us to use a branch-and-bound approach to obtain an optimal solution in a reasonable time for practical problems.

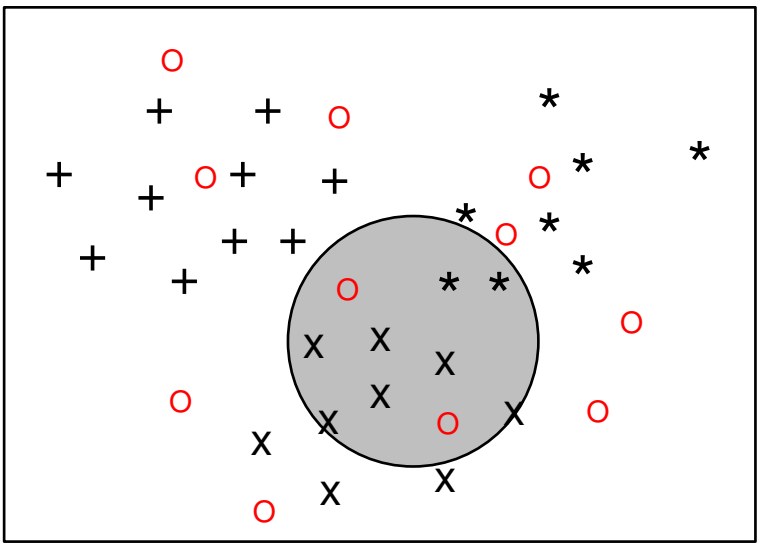


Figure 3.1. Spatially coherent sampling.

3.3 Branch-and-Bound Algorithm for Segmentation

This section constructs the branch-and-bound algorithm for the optimization of cost function in (3.6). A branch-and-bound algorithm requires formulation of various components such as branching, bounding, pruning and retracting [43]. Apart from bounding, all the other components can be represented as a rooted tree. In the following subsection, the tree representation of the MSaM problem is formulated.

3.3.1 Solution Tree

There are N_h hypotheses $H = \{\mathtt{F}_1, \mathtt{F}_2, \ldots, \mathtt{F}_{N_h}\}$ for the fundamental matrices $\mathtt{F}_j$ and we have to choose K of them to minimize the criterion in (3.6). All possible solutions of this optimization problem can be represented as a rooted tree. Each node encountered on the tree represents a solution. The node is also the partial solution for its descendent nodes. It is important that every solution is listed only once to avoid unnecessary computations. This can be ensured by creating child nodes that are different than:

- left siblings,
- ancestors,
- left siblings of ancestors.

One simple way of generating such a solution tree for $N_h = 5$ is shown in figure 3.2 with an additional null hypothesis $\mathtt{F}_0$ shown as $z_0 = 0$ in the figure. The null hypothesis is that for a given match, none of the N_h hypotheses is valid. This means that the match is an outlier. We will introduce the null hypothesis in detail later in the section. Note that, in the solution tree ($z_0 < z_1 < z_2 < z_3 \ldots < z_n$) and (left sibling < right sibling). These two conditions ensure that the rule stated above to generate the child nodes is followed. Note that this gives rise to a binomial tree of degree N_h [65]. This tree has the following properties.

- It has 2^{N_h} nodes and each node corresponds to a solution.
- The height of the tree is N_h, which is equal to the largest possible value of K.
- At any given depth z_n, it has $\frac{z_n!}{N_h!(N_h-k)!}$ nodes.

The solution tree can be explored by search algorithms such as breadth first search and depth first search. We choose the depth first search to take advantage of the recursive relationships of various computations, which will be clear in our discussion later. In general, depth first search avoids the exponential space complexity as well. Any tree search algorithm is applied as a series of *branch forward*, *branch right* and *retraction* operations [43].

To understand the various tree operations and their physical interpretation, we assume that the circled node in figure 3.2 indicates the current search location. The current location can be represented by the nodes traversed to reach it, i.e. $\langle 0, 1, 3 \rangle$. This means that the null hypothesis, hypotheses $\mathtt{F}_1$ and $\mathtt{F}_3$ are included in the current solution. We can define a binary representation for the node. In an N_h bit wide binary representation, the nodes traversed to reach the current node can be represented by

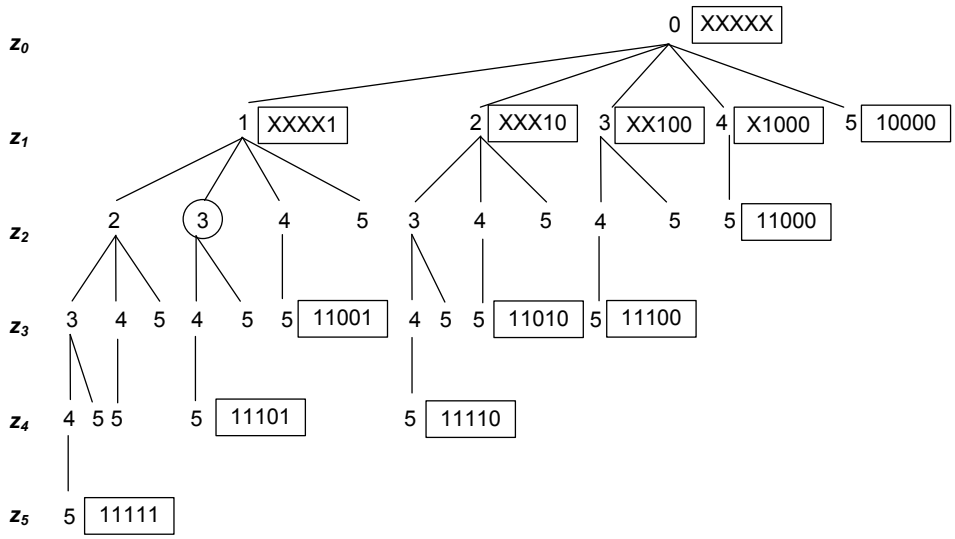


Figure 3.2. Solution tree for $N_h = 5$ and a null hypothesis, number in the rectangle indicates extended representation for the node.

'1' and the hypotheses that are not traversed can be indicated by '0'. With a slight abuse of notation, we extend this binary notation to include representation for the partial solutions. We represent the hypotheses that can be traversed in the future by X (don't care). By replacing Xs by '0's, the *extended* partial solution representation of a node can be turned into a *binary* solution representation of the same. Thus, the circled node in figure 3.2 is represented by XX101. The extended representation not only indicates that the current solution is 00101 but through Xs it also indicates that it is the partial solution for solutions 01101 i.e. $\langle 0, 1, 3, 4\rangle$, 10101 i.e. $\langle 0, 1, 3, 5\rangle$ and 11101 i.e. $\langle 0, 1, 3, 4, 5\rangle$.

A branch forward operation moves deeper in the tree by one level. After a branch forward operation, the partial solution $\langle 0, 1, 3\rangle$ would lead to $\langle 0, 1, 3, 4\rangle$. In terms of the extended representation, the trailing X is replaced by '1'. A branch forward operation adds one more hypothesis to the solution. A branch right operation moves to the sibling branch towards right. In the extended representation, leading '1'

is replaced by '0' and trailing X is replaced by '1'. The solution $\langle 0, 1, 3\rangle$ would branch right to give the solution $\langle 0, 1, 4\rangle$. A branch right operation replaces a hypothesis with the next hypothesis. Thus, the number of hypotheses after the branch right operation remains the same. A retraction moves the solution one level up the tree. A retraction is carried out when no forward or right branching is possible. Solution $\langle 0, 1\rangle$ is the result of the retraction at the circled node. For the extended representation, the operation first replaces leading 1 with X and then all the leading '0's with X. Note that a retraction is generally followed by a branch right step. The branch-and-bound algorithm is terminated when a retraction leads to the root node.

3.3.2 Monotonicity of Partial Costs

The solution representing a node at depth n be given by

$$Z(n) = \{\langle z_0, z_1, z_2, \ldots, z_{n-1}, z_n\rangle, \mathcal{D}(n)\}.$$

The hypotheses $z_0, z_1, z_2, \ldots, z_{n-1}, z_n$ correspond to fundamental matrices

$$\mathtt{F}_{z_0}, \mathtt{F}_{z_1}, \mathtt{F}_{z_2}, \ldots, \mathtt{F}_{z_{n-1}}, \mathtt{F}_{z_n}$$

respectively. Minimum distances at depth n are given by

$$\mathcal{D}(n) = [D(1, n), D(2, n), \ldots, D(N, n)]$$

and $D(i, n)$ corresponds to the minimum distance for the ith match among the current set of hypotheses.

$$D(i, n) = \min_{k=0}^{n} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_k}). \tag{3.7}$$

The cost function for the solution $Z(n)$ can be written as,

$$\mathcal{C}(Z(n)) = \underbrace{\frac{1}{\sigma^2}\sum_{i=1}^{N} D(i,n)}_{\text{Negative log likelihood},\tilde{L}} + \underbrace{\alpha \cdot n}_{\text{Penalty}}, \tag{3.8}$$

The cost function is made up of two terms, one corresponding to the negative log likelihood $\tilde{L}$ and another corresponding to penalty.

Eqn. (3.7) can be rewritten as,

$$\begin{aligned} D(i,n) &= \min\left\{\min_{k=0}^{n-1} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_k}), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_n})\right\} \\ &= \min\left\{D(i,n-1), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_n})\right\}. \end{aligned} \tag{3.9}$$

Thus, in terms of the partial solution $D(i, n-1)$, the newly formed $D(i,n)$ can be calculated incrementally with (3.9). When a new hypothesis z_n is added to the existing partial solution, it means that a new cluster center $\mathtt{F}_{z_k}$ is being added. The matches, which are close to the new cluster center, are reassigned to the new cluster, while others remain unchanged. Additionally, it is clear from (3.9) that,

$$\forall i, D(i,n) \leq D(i,n-1).$$

This suggests that,

$$\sum_{i=1}^{N} D(i,n) \leq \sum_{i=1}^{N} D(i,n-1) \tag{3.10}$$

$$\tilde{L}\{Z(n)\} \leq \tilde{L}\{Z(n-1)\}. \tag{3.11}$$

When a new cluster center is added, the penalty term of cost function increases by α while the negative log likelihood term decreases or remains the same. We will

use these monotonicity properties in the following subsection to establish the lower bound on the cost function.

Leading from the monotonic decrease of the negative log likelihood and the linear increase of penalty term, a monotonicity requirement can be imposed on the optimal solution. We define *likelihood value* for a hypothesis z_m, $G_n(z_m)$ as increase in the negative log likelihood of the solution $Z(n)$, if z_m is removed from the solution to form a new solution $Z'(n-1)$ [Note that $Z'(n-1)$ and $Z(n-1)$ are different if $m \neq n$]. Similar to the likelihood value $G_n(z_m)$, one can also define $v_n(i, z_m)$, the *per pixel value* of hypothesis z_m at depth n for pixel i as,

$$v_n(i, z_m) = D'(i, n-1) - D(i, n) \tag{3.12}$$

Where $D'(i, n-1)$ is the minimum distance for the ith match with an updated set of hypotheses $(z_0, z_1, z_2, \ldots, z_{n-1}, z_n) \setminus z_m$. From the definition, the likelihood value can be written in terms of the per pixel value of a hypothesis as,

$$G_n(z_m) = \frac{1}{\sigma^2} \sum_{i=1}^{N} v_n(i, z_m). \tag{3.13}$$

Using these definitions, we construct the proof of the monotonicity of the cost function.

Theorem 3.3.1 *The per pixel value of a hypothesis z_m for pixel i is largest when it is first added, i.e. for any $n > m$, $v_n(i, z_m) \leq v_m(i, z_m)$.*

Proof A hypothesis z_m is first added at depth m. For depth $n < m$, z_m is not part of the solution and has zero per pixel value. From (3.7) and (3.12), the per pixel value of hypothesis z_m for pixel i when $n \geq m$,

$$\begin{aligned} v_n(i, z_m) &= \min \left\{ \min_{k'=0}^{m-1} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_{k'}}), \min_{k''=m+1}^{n} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_{k''}}) \right\} \\ &\quad - \min_{k=0}^{n} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_k}) \qquad (3.14) \\ &= \left(\min \left\{ \min_{k'=0}^{m-1} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_{k'}}), \min_{k''=m+1}^{n} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_{k''}}) \right\} \right. \\ &\quad \left. - d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_m}) \right)_+ \qquad (3.15) \end{aligned}$$

where,

$$(f(\cdot))_+ = \begin{cases} f(\cdot), & \text{if } f(\cdot) > 0; \\ 0, & \text{otherwise.} \end{cases}$$

is a function, which maps negative values to zero, while keeping positive values unchanged. When $n = m$

$$v_m(i, z_m) = \left(\min_{k'=0}^{m-1} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_{k'}}) - d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_m}) \right)_+ . \qquad (3.16)$$

Comparing (3.15) and (3.16), for any $n > m$

$$v_n(i, z_m) \leq v_m(i, z_m). \qquad (3.17)$$

Theorem 3.3.2 *For optimality of a solution $Z(n)$, it is necessary that $\alpha \leq G_n(z_m)$ for all $m \leq n$.*

Proof If the solution $Z(n)$ is optimal then for any $m \leq n$,

$$\begin{aligned} \mathcal{C}\{Z(n)\} &\leq \mathcal{C}\{Z'(n-1)\} \\ \tilde{L}\{Z(n)\} + \alpha \cdot n &\leq \tilde{L}\{Z'(n-1)\} + \alpha \cdot (n-1) \\ \alpha &\leq \tilde{L}\{Z'(n-1)\} - \tilde{L}\{Z(n)\} \\ \alpha &\leq G_n(z_m) \end{aligned} \tag{3.18}$$

Theorem 3.3.3 *If the initial likelihood value of a hypothesis z_m, $G_m(z_m) < \alpha$, all the solutions leading from the current partial solution cannot be optimal.*

Proof From (3.13) and (3.17),

$$G_n(z_m) \leq G_m(z_m). \tag{3.19}$$

If $G_m(z_m) < \alpha$ then,

$$G_n(z_m) < \alpha. \tag{3.20}$$

Then according to Theorem 3.3.2, any solution, which includes z_m, cannot be optimal.

If $\mathcal{C}\{Z(n)\} > \mathcal{C}\{Z(n-1)\}$ then $G_m(z_m) < \alpha$. Thus, for Theorem 3.3.3 to hold, the cost function must be monotonically decreasing.

3.3.3 Lower Bound on Cost

To establish a lower bound on the cost, we define a complementary variable $D^*(i, z_n)$ as,

$$D^*(i, z_n) = \min_{k=z_n+1}^{N_h} d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_k).$$

The variable $D^*(i, z_n)$ gives the minimum of the distance measures from all hypotheses, which can be included in the solution in the future. In case of the variable $D^*(i, z_n)$, its value solely depends on the last node z_n. As there are only N_h possibilities for the value of z_n, $D^*(i, z_n)$ can be pre-computed to speed up the branch-and-bound process. Similar to $D(i, n)$, $D^*(i, n)$ can also be calculated incrementally as,

$$D^*(i, z_n) = \min[D^*(i, z_n + 1), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{z_n+1})].$$

Consider a possible partial solution $Z(4) = \langle 0, 1, 3, 4, 7 \rangle$ for $N_h = 10$. The variable D at level $n = 4$ can be computed as,

$$\begin{aligned} D(i, 4) \quad = \quad & \min\{d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_0), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_1), \\ & d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_3), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_4), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_7)\}. \end{aligned}$$

Now for the same example, we consider the complementary variable D^*.

$$D^*(i, \mathtt{F}_7) \quad = \quad \min\{d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_8), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_9), d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_{10})\}.$$

With the help of the complementary variable, the lower bound on the solutions leading from $Z(n)$ is,

$$\begin{aligned} \mathcal{C}_{\text{Lower}}(Z(n)) = \frac{1}{\sigma^2} \sum_{i=1}^{N} & \min[D(i, n), D^*(i, z_n)] \\ & + \alpha \cdot (n + 1). \end{aligned} \tag{3.21}$$

If $\mathcal{C}_{\text{Lower}}(Z(n)) > \mathcal{C}^*$, then the current partial solution can be safely abandoned as it would not lead to a better solution than the current optimal solution $\mathcal{C}^*$. Figure 3.3 pictorially represents the computation of the bound. Each stack of parallelograms

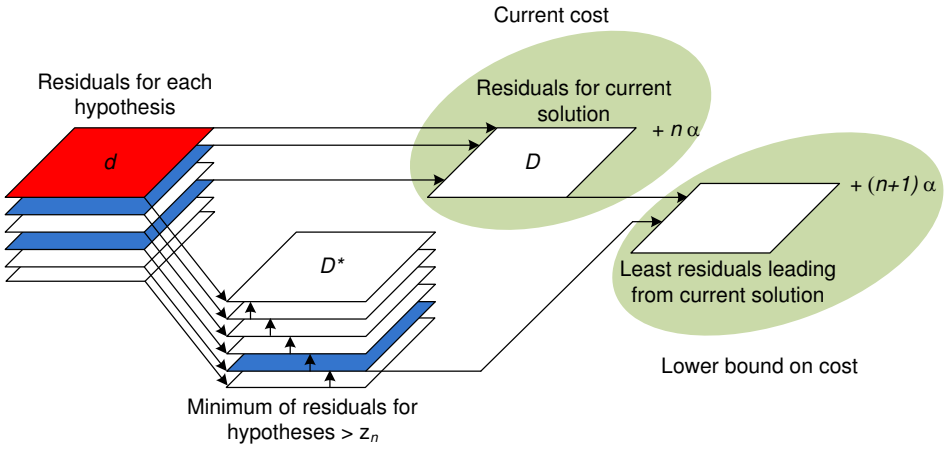


Figure 3.3. Computation of lower bound on cost.

indicates various quantities involved in bound computation and arrow-heads leading to a parallelogram indicate minimum taken over parallelograms attached to the arrow-tails.

3.3.4 Null Hypothesis Likelihood

Matching errors are common in the MSaM problems. These errors can severely deteriorate the quality of the solutions achieved for the MSaM segmentation. The outliers can be assumed to be uniformly distributed throughout the image with likelihood d_0. For ease of notation, we assume that,

$$d_0 = d(\mathbf{x}_i, \mathbf{x}'_i, \mathtt{F}_0).$$

With the introduction of this outlier likelihood as null hypothesis, the proposed MSaM segmentation scheme would act as a simple redescending M-estimator [66].

3.3.5 Branch-and-Bound Algorithm

Based on the monotonicity requirement and the lower bound, the branch-and-bound segmentation algorithm is listed below.

1. Initialization: Set the tree level $n = 1$, current node $z_0 = 0$ and current optimal cost $\mathcal{C}^* = \mathcal{C}(Z(0))$.
2. Generate child nodes: Initialize $LIST(n)$,

$$List(n) = \{z_{n-1} + 1, z_{n-1} + 2, \ldots, N_h\}$$

3. Select new node: If $List(i)$ is empty, to step (5). Otherwise, set $z_n = k$ where $k \in List(i)$. Set current solution $Z(n) = \{z_0, z_1, \ldots, z_n\}$. Delete k from $List(i)$.
4. Check bounds:
   - Compute $\mathcal{C}(Z(n))$ and $\mathcal{C}_{\text{Lower}}(Z(n))$.
   - If $\mathcal{C}(Z(n)) < \mathcal{C}^*$, set $\mathcal{C}^* = \mathcal{C}(Z(n))$ and $Z^* = Z(n)$.
   - If $\mathcal{C}(Z(n-1)) < \mathcal{C}(Z(n))$ or $\mathcal{C}_{\text{Lower}}(Z(n)) > \mathcal{C}^*$, go to step (3).
   - If $\mathcal{C}(Z(n-1)) > \mathcal{C}(Z(n))$ and $\mathcal{C}_{\text{Lower}}(Z(n)) < \mathcal{C}^*$, set $i = i + 1$ and go to step (2).
5. Backtrack to lower level: Set $n = n - 1$. if $n > 0$ go to step (3), otherwise terminate the algorithm.

The flowchart of the algorithm is shown in figure 3.4. In the following section, the branch-and-bound hypothesis selection was implemented and the achieved results are presented.

3.4 Experimental Results

The proposed MSaM segmentation approach was implemented and tested with synthetic as well as publicly available data sets. The MSaM segmentation was im-

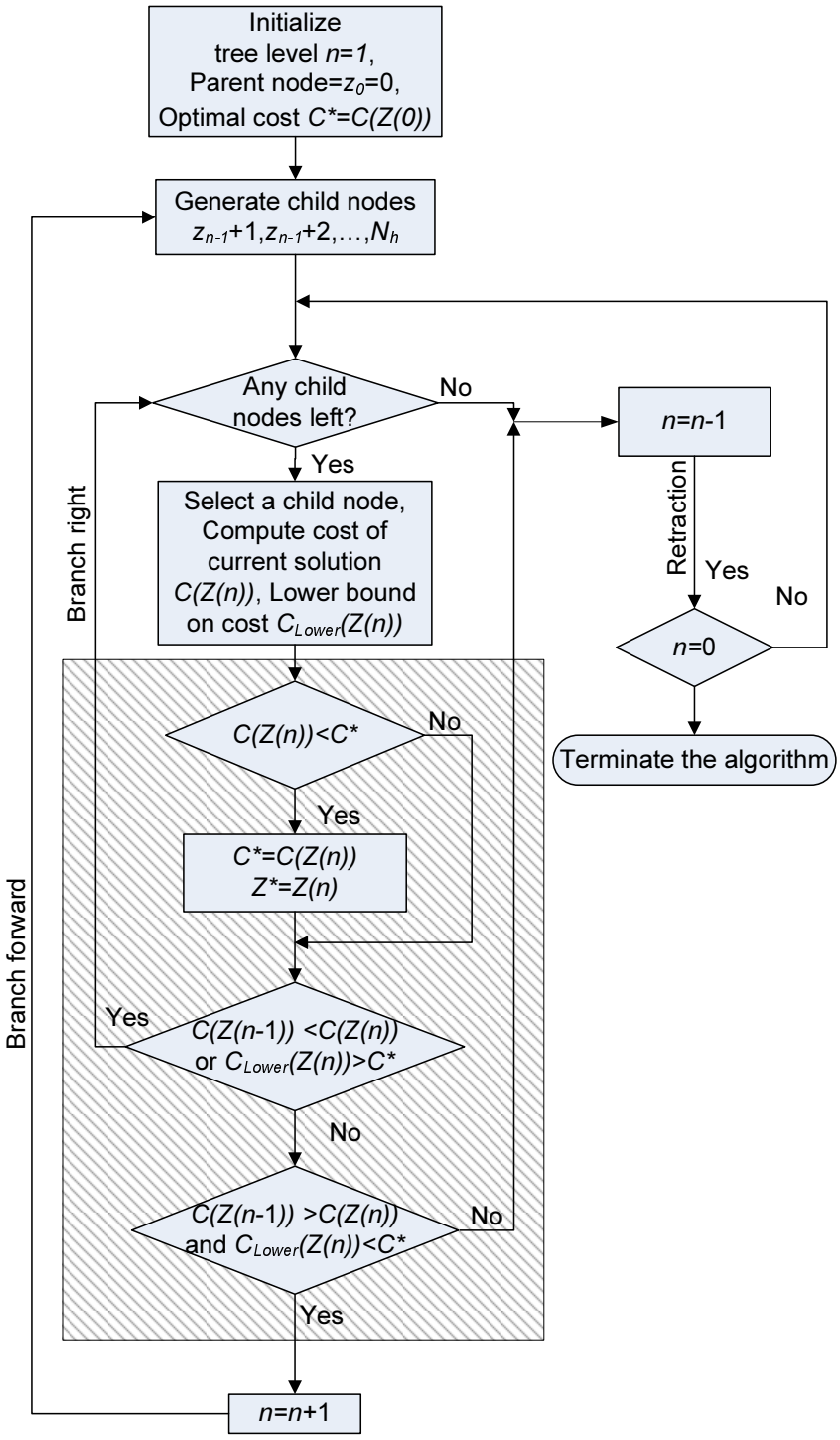


Figure 3.4. Flowchart of the proposed algorithm, hashed portion of the chart checks for various bounds.

plemented in Matlab and executed on a Core 2 Duo processor operating at 2.33GHz as a single thread.

To generate the motion hypotheses, for each matched image feature the fundamental matrix was computed from its circular neighborhood. Matches in the neighborhood were used to compute the fundamental matrix using "Structure-and-Motion Toolkit" from [67]. Similar to RANSAC, outliers and inliers were selected for each fundamental matrix with d_0 as the threshold. To avoid repeated hypothesis, which are similar, hypothesis with smaller support which had substantial ($> 80\%$) overlapping inliers with larger hypothesis were suppressed. Finally, the surviving hypotheses were arranged in a decreasing order of the number of inliers. The BIC was optimized for these hypotheses to select the optimal hypotheses combination.

3.4.1 Synthetic Data

The proposed MSaM segmentation approach was first tested with synthetic data. For the experiments, 100 random 3D motions were generated with [67][1]. These motions were combined together to form various experimental data sets. The goal of these experiments was to test effectiveness of the approach for clusters with varying cluster size and varying number of clusters. The results for the experiments are shown in figure 3.5 and are discusses in the rest of this subsection.

Four different sets of experiments were carried out. The experimental results are presented as cluster detection accuracy and classification accuracy for each of the set.

[1]We use the function 'torr_gen_2view_matches' with the default parameters.

3.4.1.1 50 Outliers, 1 Cluster of Varying Size 10 to 50

For the first experiment in this set, one of the 100 motions was randomly selected and 10 samples of the motion were selected. One sample each from 50 other motions was selected to form the outlier cluster. The MSaM segmentation was carried out to calculate the number of clusters and the cluster memberships. This process was repeated 100 times. The experiment was repeated by changing the size of the inlier cluster to 20 (experiment 2), 30 (experiment 3), 40 (experiment 4) and 50 (experiment 5). For these set of experiments, the expected number of clusters was 2, one for the outliers and one for the motion with varying number of samples. Since the framework detects at least one inlier cluster, as seen in figure 3.5(a) 2 clusters are always detected irrespective of the varying inlier cluster size, which leads to a 100% cluster detection accuracy for all the experiments. Thus, the cluster detection accuracy is invalid for this experiment. However, it should be noted that the number of clusters is almost never overestimated.

The outliers were also included in estimating the classification accuracy, i.e. to reach 100% accuracy all the inliers must be labeled as one cluster while all the outliers should be labeled as the other cluster. It can be seen that for the inlier cluster size of 10 the classification accuracy is 79.1%, which indicates that majority of 83.33% outliers are correctly identified as outliers. As the varying cluster size goes to 20 and beyond, the classification accuracy is greater than 94%.

3.4.1.2 50 Outliers, 1 Cluster of Size 50, 1 Cluster of Varying Size 10 to 50

The second set of experiments was carried out by adding a randomly selected motion with sample size 50 to the data in the first experiment. Thus, the expected number of clusters was 3 in this experiment; one for the outliers, one for the inlier

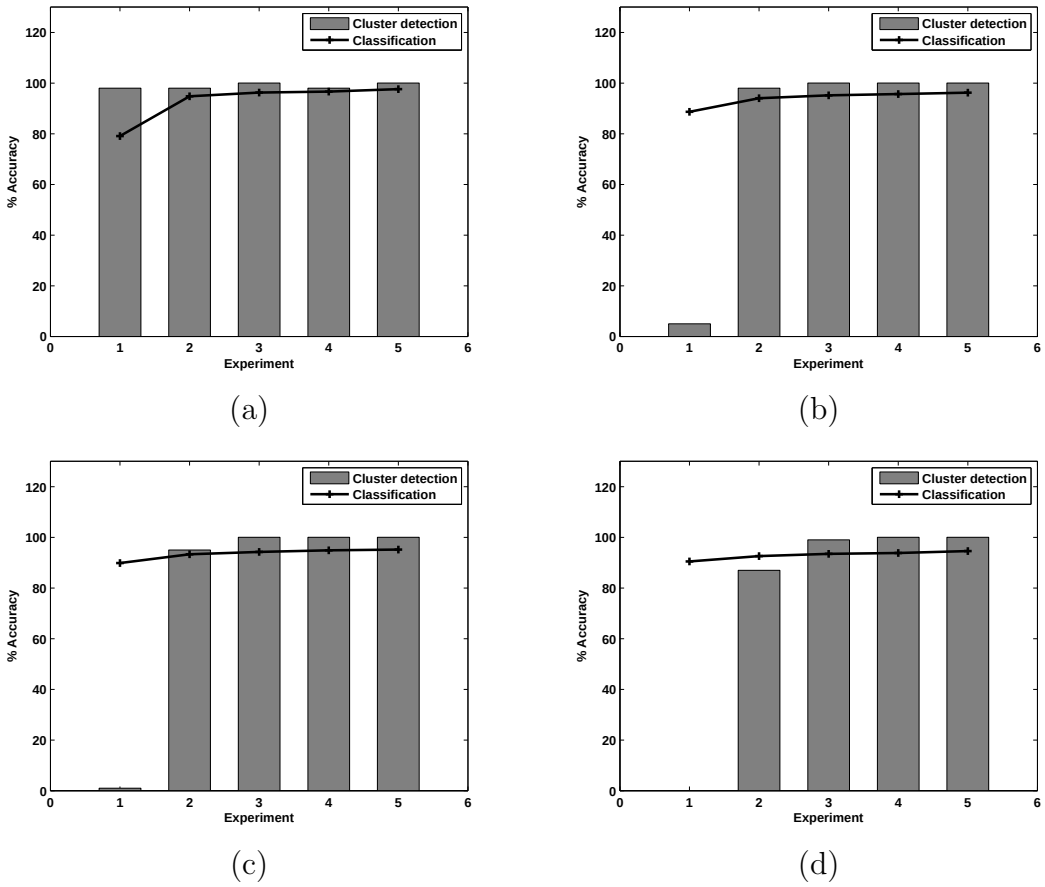


Figure 3.5. Synthetic data cluster detection and classification accuracy (a) Set 1 - 50 Outliers + 1 cluster of varying size 10 to 50, (b) Set 2 - 50 Outliers + 1 cluster of size 50 + 1 cluster of varying size 10 to 50, (c) Set 3 - 50 Outliers + 2 clusters of size 50 each + 1 cluster of varying size 10 to 50, (d) Set 4 - 50 Outliers + 3 clusters of size 50 each + 1 cluster of varying size 10 to 50.

motion of sample size 50 and one for the inlier motion with varying cluster size. When varying cluster size is 10 (experiment 1), our method fails to detect that cluster 95% of times (figure 3.5(b)). This happens as it is hard to obtain a clean sample to detect the inlier hypothesis due to the large number of outliers compared to the inliers. Additionally, at times for less number of samples it might be cheaper to explain them as outliers rather than assigning them a new cluster. In this scenario, the expected classification accuracy is $90.9\%((50+50)/(50+50+10))$ if all the outliers

and the inlier cluster of size 50 is correctly identified. The experimental classification accuracy is 88.67%, which is close to the expected accuracy. When the varying cluster size becomes 20 (experiment 2), the cluster detection accuracy is 98%. As the size of the cluster goes beyond 30 (experiments 3, 4 and 5) the cluster detection accuracy reaches almost 100% and the classification accuracy percentage reaches about 95.

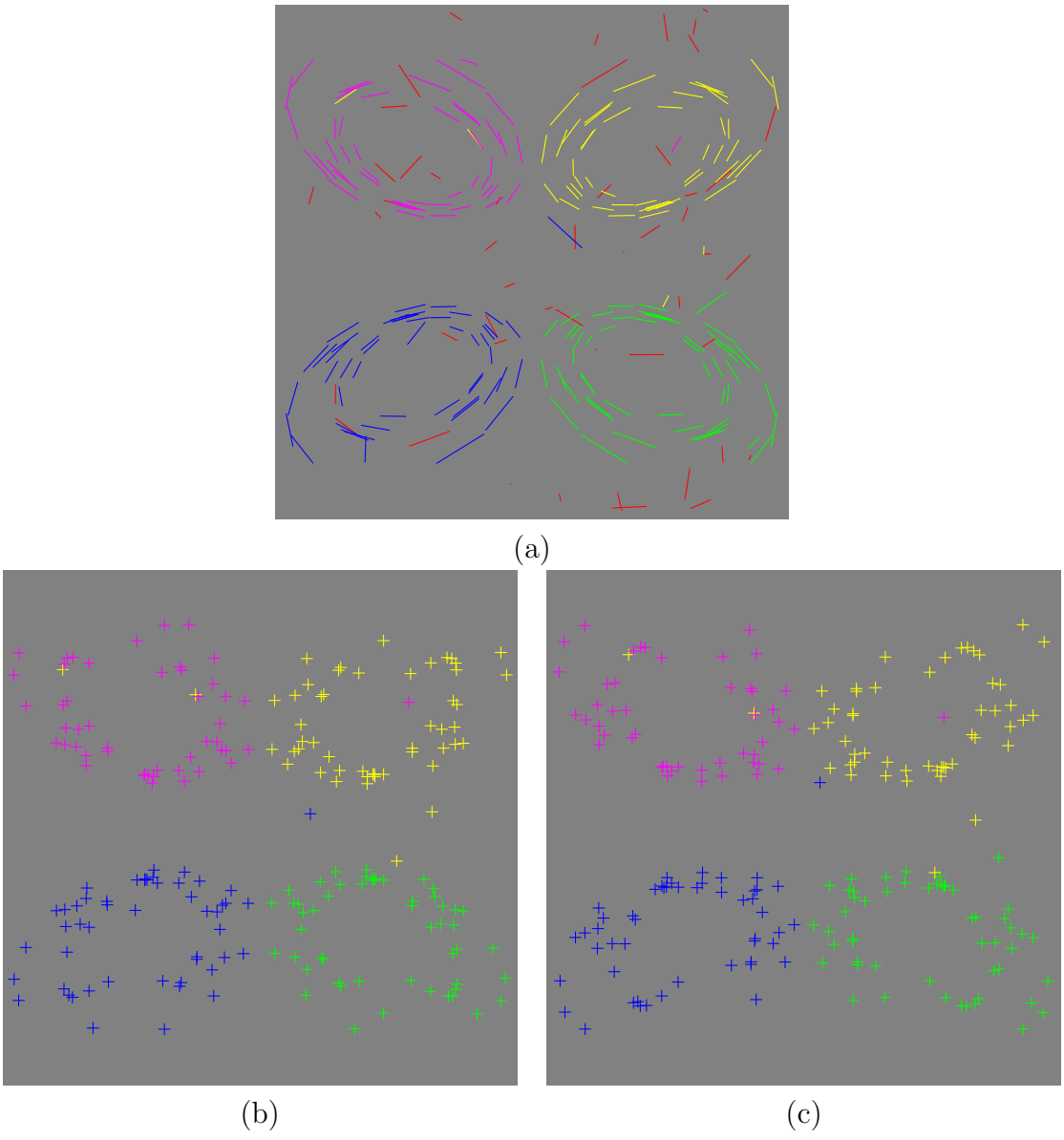
3.4.1.3 50 Outliers, 2 Clusters of Size 50 each, 1 Cluster of Varying Size 10 to 50

The third set of experiments was carried out by adding a randomly selected motion with sample size 50 to the data in the second experiment. The expected number of clusters is 4 in these experiments. Failure to detect the motions, which have 10 samples, continues in this set of experiments. However, the cluster detection accuracy drops to 95% for the cluster size of 20. This happens because as the number of data points becomes larger, adding a cluster becomes more expensive.

3.4.1.4 50 Outliers, 3 Clusters of Size 50 each, 1 Cluster of Varying Size 10 to 50

The third set of experiments was carried out by adding a randomly selected motion with sample size 50 to the data in the third experiment. The cluster detection accuracy as well as classification accuracy in this case is slightly lower compared to previous experiments. However, this is expected due to increase in clustering penalty.

In another synthetic data experiment, we use "Spinning wheels" synthetic test data from [68]. This sequence contains four rotating objects with 50 tracked points each with 50 outliers. Frame 1 and 3 of the sequence were used in our experiment. After sampling and non maximal suppression, 22 hypotheses were selected. The proposed approach detects 4 clusters along with outliers. The total number of solutions explored by the branch-and-bound process was 2043.



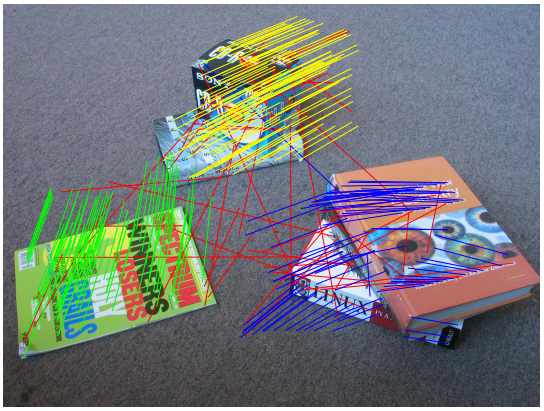
(a)

(b) (c)

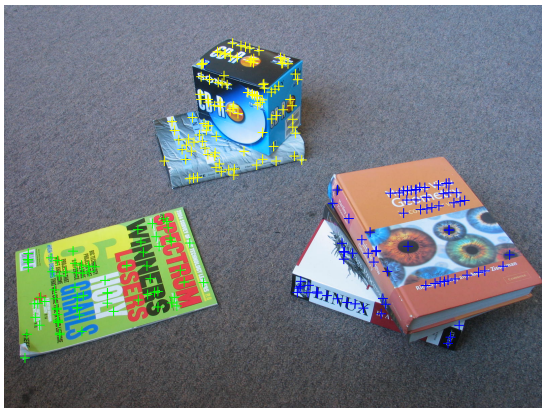
Figure 3.6. Spinning wheels: (a) Disparities between two frames, each cluster is denoted by different color, matches marked by red are outliers; (b) Segmentation result for the first frame; (c) Segmentation result for the second frame.

3.4.2 Real Data

For all the real data used in these experiments, we use sparsely matched features provided by the provider of the data. For the first experiment with real data, "Box-book-mag" and "Desk" image pairs from [69] are used. "Box-book-mag" pair has three independently moving objects while camera is stationary. Figure 3.7 (a) shows disparities between the image pair indicated in different colors. While red colored matches are the detected outliers, each of rest of the colors represents disparities for a segmented object.



(a)

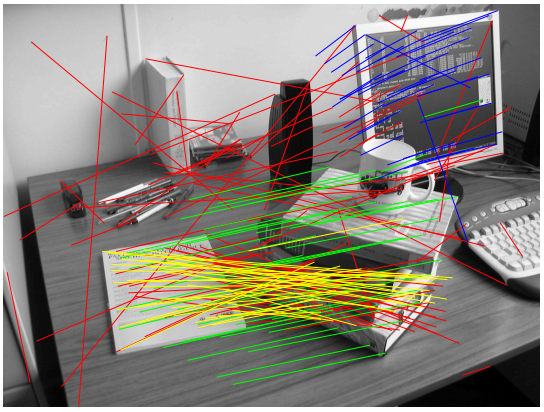


(b)

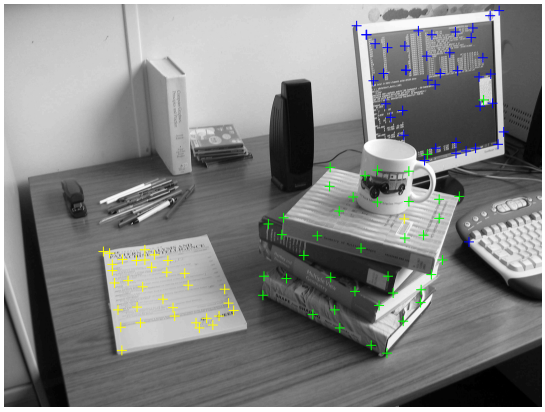
(c)

Figure 3.7. Box-book-mag: (a) Disparities between two views, each cluster is denoted by different color, matches marked by red are outliers; (b) Segmentation result for the first view; (c) Segmentation result for the second view..

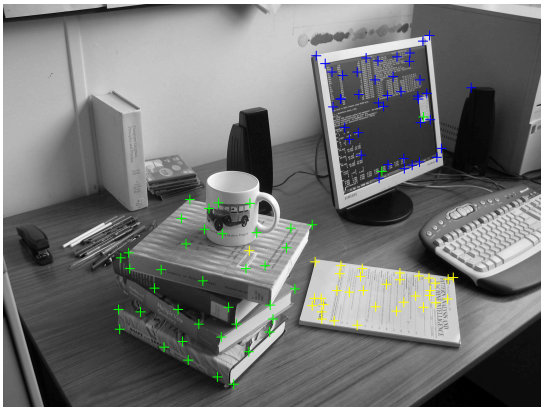
For the "Desk" image pair shown in figure 3.8, there are three moving objects namely the pile of books, the computer screen and the journal. Although the camera has also moved, there are no matches for the background. Thus, the background motion is not detected. The result of segmentation can be seen in figure 3.8 (b) and (c).



(a)



(b)

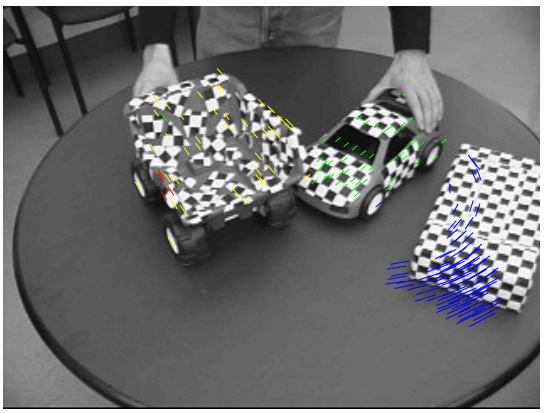


(c)

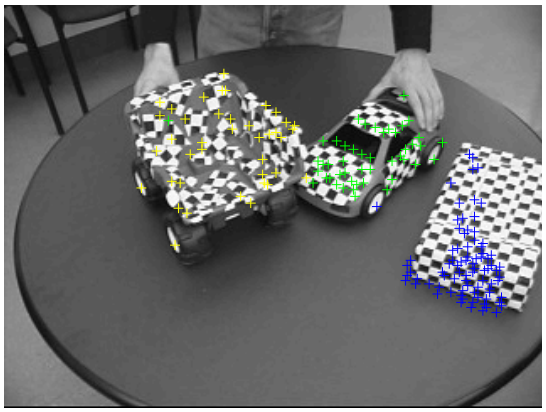
Figure 3.8. Desk: (a) Disparities between two views, each cluster is denoted by different color, matches marked by red are outliers; (b) Segmentation result for the first view; (c) Segmentation result for the second view.

In the next experiment, our method is applied to the "car-truck-box" sequence used by Vidal *et al.* [20,70]. The motion between frame 1 and frame 8 of the sequence

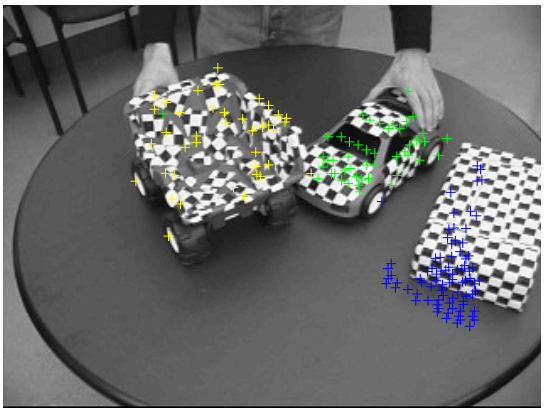
was analyzed. In this sequence, there are three different motions. The box lies on a rotating desk, while the car and the truck are moved away from each other with hand. As seen in 3.9, three moving objects are correctly identified; however some of the motion vectors are incorrectly assigned. This is due to the sampling scheme that we use, rather than the cost function being optimized. If optimal motions are subset of the hypotheses being constructed then the segmentation results are guaranteed to be optimal with respect to the cost function.



(a)



(b)



(c)

Figure 3.9. Car-truck-box: (a) Disparities between two views, each cluster is denoted by different color, outliers are marked by red; (b) Segmentation result for frame 1; (c) Segmentation result for frame 8.

In the next sequence, taken from Sugaya and Kanatani [71] has a single moving object, i.e. the car. However, camera is also moving for this sequence. Frame 10 and frame 15 are used for segmentation in our experiment. The egomotion of camera and the motion of the car are correctly segmented and are shown in figure 3.10(c).

(a)

(b)

(c)

Figure 3.10. Kanatani: (a) Disparities between two views, each cluster is denoted by different color; (b) Segmentation result for frame 10; (c) Segmentation result for frame 15.

Finally, the proposed approach was tested with JHU155 database sequences [72], which includes various checkerboard and traffic sequences with two or three motion groups. The “cars3” sequence shown in Fig. 3.11 (a) has two moving cars

captured by a moving camera. Fig. 3.11 (b) gives segmentation results for the "people1" sequence, which depicts a pedestrian captured by a moving camera. The "truck2" sequence is segmented in the moving vehicle and the background in Fig. 3.11 (c). The checkerboard sequence in Fig. 3.11 (d), with one rotating object and one translating object captured by a rotating camera, was also successfully segmented by the proposed approach.

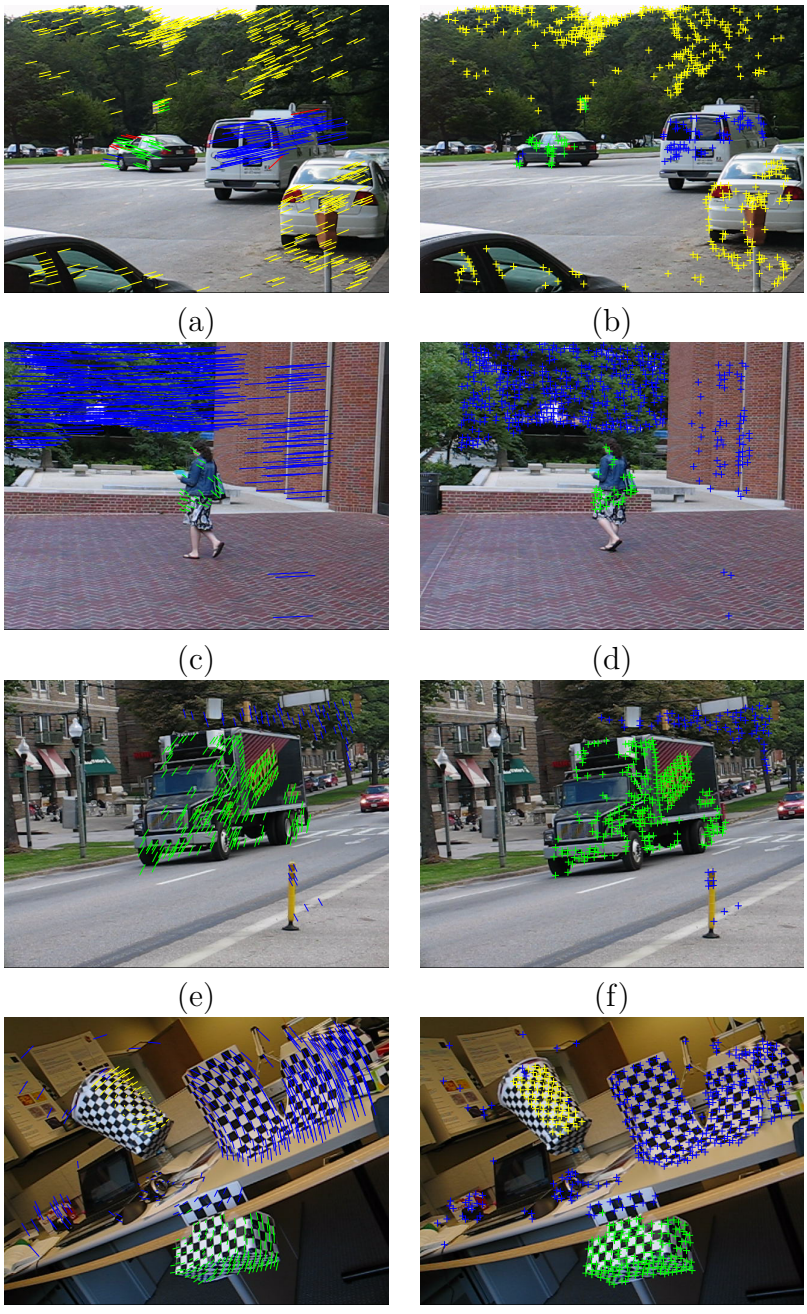
Table 3.1 shows summary of the execution of our method for all the experiments. Fraction of solutions explored shown in the table is calculated as,

$$\text{Fraction explored} = \frac{\text{Solutions explored}}{2^{N_h}}.$$

As seen for the table, the fraction of all the solutions explored is very small. This is also reflected in the execution speed. Note that the execution times for search alone are listed and they do not include sampling and pre-computing involved. Speedups achieved increase with increase in N_h since more solutions are generally rejected implicitly by rejecting a partial solution.

Table 3.1. Execution summary for the experiments

Sequence	N_h	Solutions explored	Fraction explored	Time (Seconds)
Spinning wheels	22	2043	4.87e-4	0.17
Three car-Vidal	35	30542	8.89e-7	2.54
Book-box-mag	16	511	7.80e-3	0.06
Desk	25	1898	5.66e-5	0.16
Kanatani	14	354	2.16e-2	0.06
cars3-JHU	53	20166	2.24e-12	2.44
people1-JHU	95	7149	1.80e-25	0.94
truck2-JHU	34	622	3.62e-8	0.07
1R2TCR-JHU	58	21550	7.48e-14	2.40



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 3.11. Sequences from JHU155 database: Left: Segmentation result with disparities for the first view, Right: Segmentation result for the second view (a)(b) “cars3” sequence; (c)(d) “people1” sequence; (e)(f) “truck2” sequence; (g)(h) “1R2TCR” sequence.

In the following chapter, we generalize the segmentation framework presented here and establish its average computational complexity.

CHAPTER 4

COMPUTATIONAL COMPLEXITY OF BRANCH-AND-BOUND

4.1 Introduction

Clustering is a popular unsupervised learning technique applied in areas such as data mining [73], image processing [74], pattern recognition [75] and bioinformatics [76]. It meaningfully organizes the data by grouping similar data points in to a cluster and splits dissimilar data points in to different clusters. Normally, the similarity between data points is assessed with the help of a dissimilarity or distance measure such as Euclidian distance. The classical clustering method of k-means divides the data in to k partitions so that the sum of squared error between cluster means and the data in the corresponding cluster is minimized. The k-means procedure falls under the category of partitioning methods for clustering. Hierarchical methods of clustering create a hierarchy of clusters.

In an agglomerative hierarchy, smaller clusters are merged to construct larger clusters, starting from individual data points leading to a single cluster. Under a divisive hierarchy, larger clusters are divided to form smaller clusters. The divisive strategy starts with a single cluster and finally each data point corresponds to a cluster. The desired complexity of clustering can be generated by cutting the hierarchy at a predetermined depth. Density based clustering approaches grow clusters based of the density of data points in the clustering space. Unlike partitioning approaches, the density based approaches can detect clusters of arbitrary shapes. In the model-based clustering approach, each cluster is represented by a parametric model [77]. A data point is assigned to the cluster whose model explains the data point the best.

A model such as Gaussian mixture model (GMM) or hidden Markov model (HMM) is defined *a priori* based on the domain knowledge. Han and Kamber [73] give detailed descriptions of the contemporary techniques, which follow the aforementioned clustering paradigms.

Image, motion, stereo disparity, and structure-and-motion segmentations can be expressed as model-based clustering problems. For model-based clustering problems, the cluster parameters should be known in order to assign a data point to an appropriate cluster. On the other hand, the cluster parameters can be computed only if the cluster assignments of data points are known. This "chicken-and-egg" dilemma leads to an iterative formulation for model-based clustering methods similar to an expectation maximization (EM) algorithm [78].

Clustering aims to optimize an assignment cost such as dissimilarity measure to achieve a (locally) optimal solution. If the number of clusters is increased, generally the cost for the same data reduces. The degenerate case for this occurs when one cluster corresponds to one data point and the corresponding clustering cost is zero. Clearly, such a scenario is undesirable. Thus, the clustering cost must be appropriately adjusted, which results in penalty for additional clusters. Several model selection methods exist, which incorporate this idea [79]. Note that the term "model" in model selection refers to the number of clusters and the parametric models for these clusters. To apply model selection to clustering, candidate models are sequentially generated by varying the number of clusters and, the best model is selected according to a model selection criterion. The iterative and sequential problem of model selection for the image data can be simplified to a one step optimization by using the knowledge that the clusters formed in an image are spatially coherent. The candidates for cluster parameters can be generated by sampling spatially coherent image data points. Once the candidates are known, a subset of these candidates can

be selected by optimizing a model selection cost. This transforms the segmentation problem into a one step model selection problem.

This idea is utilized in structure-and-motion segmentation approaches proposed recently [19, 69]. Schindler and Suter [69] carry out multi-body structure-and-motion segmentation from two camera views. After correspondences are established between the two views, they are grouped together based on the spatial coherence. From each group of correspondences, a candidate hypothesis for underlying structure-and-motion model is generated using random sample consensus (RANSAC) [15]. A geometrically robust information criterion (GRIC) [80] is optimized to select the best subset of candidates hypotheses. The optimization of the criterion is carried out with Tabu search [81]. Li [19] solves the two-view motion segmentation problem starting from a set of candidate motions generated by applying spatial coherence, prior distribution etc. The segmentation problem is then formed as a facility location problem and solved with linear programming relaxation [82, 83]. We have a similar approach. In chapter 3, we generate candidates for structure-and-motion by applying local sampling followed by nonmaximal suppression. We optimize the Bayesian information criterion (BIC) [84] with a branch-and-bound strategy.

In this chapter, a general framework based on chapter 3 for multi-hypotheses branch-and-bound model selection is outlined and, its average computational complexity is analyzed. The average computational complexity of the branch-and-bound algorithms, which search over random trees has been explored by a number of researchers [85–90]. The term "random" applies to the structure of the tree and weights of the tree edges in general. However, for the multi-hypotheses branch-and-bound model selection problem, the structure of the tree is deterministic and only the weights of the tree edges are random. Thus, a separate treatment for the complexity

of the problem becomes necessary. Our preliminary work on this topic appeared in [91].

This chapter is organized as follows: Section 4.2 formulates a generalized multi-hypotheses branch-and-bound model selection problem. Section 4.3 develops the framework to estimate the expected complexity of the branch-and-bound search for the problem. The computation of various quantities involved in the estimation of complexity of the algorithm is discussed in section 4.4. Section 4.5 presents the results achieved by the model selection process and its expected complexity.

4.2 Generalized Multi-Hypotheses Branch-and-Bound Model Selection

This section first formulates the model-based clustering and model selection problem. Later, a branch-and-bound algorithm for the problem is devised and application of the model selection to an image segmentation problem is discussed.

4.2.1 Segmentation as a Model Selection Problem

Consider a set $\mathbf{Y}$ consisting M observations such as image intensity/color, video motion or stereo disparity.

$$\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\}$$

The corresponding cluster memberships for the observations can be denoted by $L = \{l_1, l_2, \ldots, l_M\}$. If an observation y_j belongs to a cluster k then $l_j = k$ and vice-versa. Under the model-based clustering paradigm, the data can be explained with one of the K clusters with parameters $\{\Theta_1, \Theta_2, \ldots, \Theta_K\}$ respectively. A generic model for

estimating observations from the cluster parameters and the memberships can be given as [92],

$$\mathbf{y}_j = g(\mathbf{x}_j; \Theta_{l_j}) + \mathbf{v}_j, \quad j = 1, 2, \ldots, M. \tag{4.1}$$

In this model, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ are the independent variables, on which observations $\mathbf{Y}$ depend (these can be quantities such as the spatial locations of the images or time instances for time-series data). If the data has no spatial or temporal relationship, which is the case for many clustering problems, the independent variables would not appear in the model [73]. $g(\mathbf{x}; \Theta_f)$ can be a linear or nonlinear function or any process that can compute observation $\mathbf{y}$ from $\mathbf{x}$ given parameters Θ_f. $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\}$ is the noise corrupting the observation, which is generally assumed to follow a zero mean independent Gaussian distribution. The model above appears in missing data problems as well [74]. According to the missing data formulation, the observations $\mathbf{Y}$ are available and the cluster memberships L are missing.

The model-based clustering problems have two unknown quantities, the cluster parameters $\Theta = \{\Theta_1, \Theta_2, \ldots, \Theta_K\}$ and the memberships L. Given the memberships L, the maximum likelihood estimate for the parameters Θ is given by

$$\hat{\Theta} = \arg\max_{\Theta} \Pr(L|\Theta, \mathbf{Y}). \tag{4.2}$$

Given the parameters Θ, the maximum likelihood estimate for the memberships L is given by

$$\hat{L} = \arg\max_{F} \Pr(\Theta|L, \mathbf{Y}). \tag{4.3}$$

After simplification,

$$\hat{\Theta}_i = \arg\min_{\Theta} \sum_{\forall j \leftrightarrow l_j = i} ||\mathbf{y}_j - g(\mathbf{x}_j; \Theta)||^2 \tag{4.4}$$

$$\hat{l}_j = \arg\min_{i} ||\mathbf{y}_j - g(\mathbf{x}_j; \Theta_i)||^2 \tag{4.5}$$

Here, $i = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, M$. Conventional methods iterate between the estimation of Θ and L till one or the other converges. They additionally require that the number of clusters K is known *a priori*. This requirement is unrealistic in most clustering problems. Thus, the number of clusters has to be varied to select the optimal number of clusters. This process is called model selection. The model selection constitutes the choice of K and the corresponding Θ. Since the likelihood of the model increases as more clusters are added, a criterion which penalizes the likelihood with increasing clusters such as Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC) is used to select the optimal number of clusters [63]. If the number of free parameters per cluster is N,

$$\mathrm{AIC}(\Theta) = -2\log(\mathcal{L}_\Theta) + 2KN. \tag{4.6}$$

$$\mathrm{BIC}(\Theta) = -2\log(\mathcal{L}_\Theta) + KN\log(M). \tag{4.7}$$

Or, a generalized model selection criterion can be given by

$$\mathcal{C}(\Theta) = -\log(\mathcal{L}_\Theta) + \alpha \cdot K, \tag{4.8}$$

where α is a positive constant and $\mathcal{L}_\Theta$ gives the likelihood of the data for a model Θ. The model selection thus leads to a sequential process, which follows the iterative clustering.

On the other hand, if a linearly ordered set of N_c candidates

$$C = \{C_1, C_2, \ldots, C_{N_c}\}$$

for cluster parameters Θ_i is given, we can choose a subset Θ, which optimizes the model selection criterion given in (4.8). The likelihood of the data is proportional to the sum of the residuals for the current model and, is given by,

$$\log(\mathcal{L}_\Theta) = -\frac{1}{2} M \log\left(\frac{\text{SSD}(\Theta)}{M}\right) + \text{Constant} \tag{4.9}$$

where,

$$\text{SSD}(\Theta) = \sum_{j=1}^{M} \min_{C_i \in \Theta} r_j(C_i)$$

and

$$r_j(C_i) = ||\mathbf{y}_j - g(\mathbf{x}_j; C_i)||^2 \tag{4.10}$$

are the residuals for jth observation for candidate C_i.

After subsuming the constants, the model selection criterion becomes,

$$\mathcal{C}(\Theta) = M \log\left(\frac{\text{SSD}(\Theta)}{M}\right) + \alpha \cdot K, \tag{4.11}$$

which is to be minimized by selecting $\Theta \subset C$, where K is the number of candidates in the subset Θ.

There are 2^{N_c} possible solutions for this subset selection problem. Even for moderate N_c, an exhaustive search is computationally expensive. However, the na-

ture of the problem allows us to use a branch-and-bound approach to obtain an optimal solution for the problem in a reasonable time for practical problems.

4.2.2 Branch-and-Bound Algorithm for Model Selection

All the possible solutions of the model selection problem can be represented by a rooted tree. It is important that every solution is listed only once in the tree to avoid unnecessary computations. This can be ensured by creating child nodes that are different than:

- left siblings,
- ancestors,
- left siblings of ancestors.

One simple way of generating such a solution tree for five candidates is shown in figure 4.1 with an additional candidate claiming that the data point is an outlier. For ease of representation, each tree node is labeled by the index of the most recently added candidate, instead of listing indices of all the candidates in the subset. The subset of candidates corresponding to a node is given by a walk from the root node to the node under consideration. The circled node includes candidates $\{C_0, C_1, C_3\}$. The candidate C_0 indicates that the data point is an outlier, i.e., the point does not belong to any of the cluster parameter candidates. Note that, in the solution tree, the node label z_i increases monotonically with the tree depth and the node label is lesser than its right sibling's label. These two conditions ensure that the rule stated above to generate the child nodes is followed.

Each node of the tree represents a subset of candidates and two hypotheses, one that *the subset gives the optimal solution* of the model selection problem and the other that *the subset is a partial solution* to the problem. A partial solution is a subset of optimal solution, i.e., adding more candidates to a partial solution would

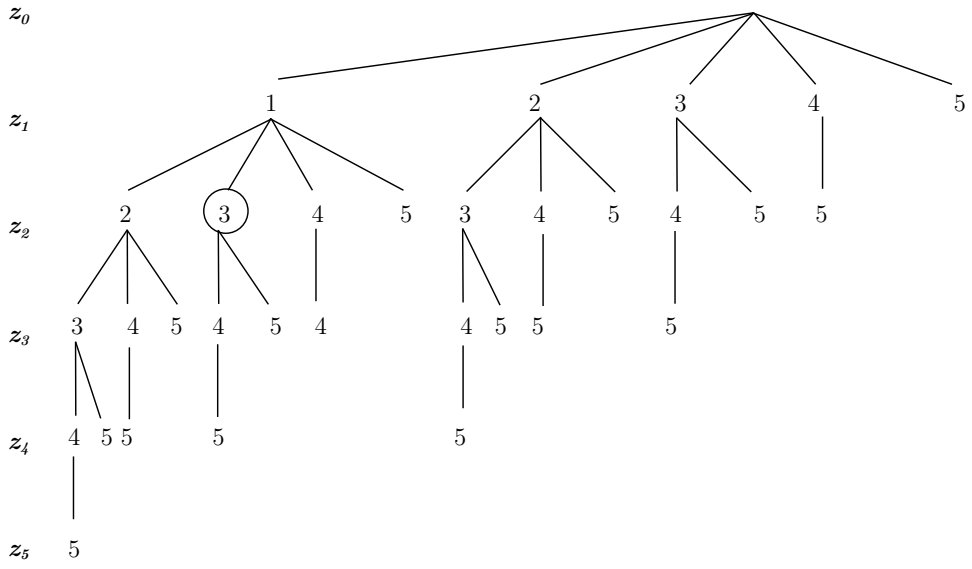


Figure 4.1. Solution tree for $N_c = 5$ and an additional candidate for outliers.

lead to an optimal solution. Note that, if a partial solution hypothesis for a node is rejected, then none of the child nodes of the node can be a partial solution. A branch-and-bound algorithm aims to validate the hypotheses presented by all the tree nodes explicitly or implicitly. As the algorithm is a search strategy, at any point of search it maintains the best search result till that point. The best search result, which is nothing but the lowest cost encountered in the search $\mathcal{C}^*$, can be used to validate the optimal solution hypothesis. If the modeling cost $\mathcal{C}(\Theta)$ for current node is higher than $\mathcal{C}^*$, the current node cannot be an optimal solution. The partial solution hypothesis can be validated with a lower bound on the modeling cost of all the solutions leading from the current subset of candidates. All the solutions leading from current subset are represented by child nodes of the current node. If the lower bound on child nodes is higher than $\mathcal{C}^*$, then the partial solution hypotheses for the node as well as its child nodes can be safely rejected.

The lower bound on the first term of (4.11) corresponds to the lower bound on SSD(Θ). The lower bound on SSD(Θ) is reached when a lower bounds on residuals

of individual data points is achieved. With candidate subset Θ, the residue for jth observation is given by,

$$\min_{C_i \in \Theta} r_j(C_i).$$

If the subset Θ is hypothesized as a partial solution, one can add a subset of candidates Θ^+ to the existing subset Θ to form a solution Θ'. To find the lower bound on the residual, one must include all the possible candidates to build Θ^+. By observing the structure of the solution tree, it is clear that the candidates $C_k \in \Theta^+$ must have $C_k > \max(\Theta)$. Note that, here the hypotheses are compared by their indices. Thus, the lower bound on the residual for jth observation is given by,

$$\begin{aligned} \min_{C_i \in \Theta'} r_j(C_i) &= \min\left(\min_{C_i \in \Theta} r_j(C_i), \min_{C_k \in \Theta^+} r_j(C_k)\right) \\ &= \min\left(\min_{C_i \in \Theta} r_j(C_i), \min_{\forall C_k > \max(\Theta)} r_j(C_k)\right). \end{aligned}$$

Thus, the lower bound on $\mathrm{SSD}(\Theta)$ is given by,

$$\mathrm{SSD}_{\mathrm{Lower}}(\Theta) = \sum_{j=1}^{M} \min\left(\min_{C_i \in \Theta} r_j(C_i), \min_{\forall C_k > \max(\Theta)} r_j(C_k)\right). \tag{4.12}$$

As at least one more candidate has to added to the partial solution of size K to reach an optimal solution, the lower bound on second term of (4.11) is given by $(K+1)$. The lower bound on the hypotheses leading from Θ is thus given by,

$$\mathcal{C}_{\mathrm{Lower}}(\Theta) = M \log\left(\frac{\mathrm{SSD}_{\mathrm{Lower}}(\Theta)}{M}\right) + \alpha \cdot (K+1) \tag{4.13}$$

From the cost function (4.11) and the bound (4.13), the branch-and-bound algorithm can be implemented.

We adapt a generic queue based implementation of the branch-and-bound procedure from [86]. With the queue based implementation, the solution tree can be explored using various search strategies. We list a few of these methods here [86],

- Best bound first (BBF)
- Ordered depth first
- Generation order depth first
- Ordered breadth first
- Generation order breadth first

These methods prioritize the search of nodes in different ways as suggested by their names. As the BBF search algorithm has the least time complexity, we choose the BBF search for our implementation and the complexity analysis. The following gives an implementation of BBF search for the model selection problem,

Best bound first branch-and-bound procedure:

1. Insert hypotheses for the root node in the priority queue Q.
2. Set the optimal cost $\mathcal{C}^* = \infty$.
3. Pop the first hypothesis from Q, which is nothing but the least cost hypothesis.
4. If the popped hypothesis is an optimal cost hypothesis then terminate the algorithm.
5. For the child nodes of the popped hypothesis, validate and insert hypotheses in Q. For all the child nodes,
   (a) Validate the optimal solution hypothesis.
      - Compute cost $\mathcal{C}(\Theta)$ for the node.
      - If $\mathcal{C}(\Theta) < \mathcal{C}^*$,
         - Insert an optimal solution hypothesis in Q with priority $1/\mathcal{C}(\Theta)$.
         - Delete hypotheses after the location where above hypothesis was inserted.

– Set $\mathcal{C}^* = \mathcal{C}(\Theta)$.

(b) Validate the partial solution hypothesis, if the node is an internal node.

- Calculate bound $\mathcal{C}_{\text{Lower}}(\Theta)$.
- If $\mathcal{C}_{\text{Lower}}(\Theta) < \mathcal{C}^*$, insert a partial solution hypothesis in Q with priority $1/\mathcal{C}_{\text{Lower}}(\Theta)$.

6. Go to step 3.

4.2.3 Application to Multiple Structure-and-Motion Segmentation

The multiple structure-and-motion (MSaM) segmentation problem groups image correspondences according to coherent structure and motion. The set of M image correspondences in this case be given by, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M\}$ and $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$, where $\mathbf{y}_j$ and $\mathbf{x}_j$ are image coordinates of the jth correspondence. If the image sequence contains K moving rigid objects, the jth image correspondence is related as,

$$\mathbf{y}_j^T \mathtt{F}_{l_j} \mathbf{x}_j = 0. \tag{4.14}$$

Here $\mathtt{F} = \{\mathtt{F}_1, \mathtt{F}_2, \ldots, \mathtt{F}_K\}$ correspond to fundamental matrices [15] of K rigid bodies. We can rewrite (4.14) as a generic model shown in section 4.2 as,

$$\mathbf{y}_j = f(\mathbf{x}_j; \mathtt{F}_{l_j}) + \mathbf{v}_j. \tag{4.15}$$

The function f here corresponds to the triangulation method [15], which can estimate $\mathbf{y}_j$ given $\mathbf{x}_j$ and the corresponding fundamental matrix. Due to the geometric nature of the problem, maximum likelihood estimation uses a geometric distance measure such as reprojection error

$$\delta(\mathbf{y}_j, \mathbf{x}_j, C_i)^2 = ||\mathbf{y}_j - \hat{\mathbf{y}}_j||^2 + ||\mathbf{x}_j - \hat{\mathbf{x}}_j||^2 \tag{4.16}$$

where $\hat{\mathbf{y}}_j$ and $\hat{\mathbf{x}}_j$ are estimated correspondences by candidate C_i given by,

$$\begin{aligned} \hat{\mathbf{y}}_j &= f(\mathbf{x}_j; C_i), \\ \hat{\mathbf{x}}_j &= f(\mathbf{y}_j; C_i^T). \end{aligned}$$

The candidates for the fundamental matrix can be generated by local sampling of the correspondences and estimating the fundamental matrix from the sample. If these candidates are $C = \{C_1, C_2, \ldots, C_{N_c}\}$, the residual for the jth correspondence for candidate C_i is given by,

$$r_j(C_i) = \delta(\mathbf{y}_j, \mathbf{x}_j, C_i)^2. \tag{4.17}$$

With these residuals, one can proceed to a generic implementation of the branch-and-bound strategy for model selection outlined at the beginning of the section.

4.3 Branch-and-Bound as an Edge-Weighted Tree Search Problem

The worst case computational complexity of any branch-and-bound search algorithm is the same as the complexity of the brute force search. However, a branch-and-bound approach is generally applied to an **NP**-hard global optimization problem, for which the worst case complexity gives little or no insight into the performance of the approach. In such a situation, the average or expected computational complexity would give a more reasonable estimate of the performance of the approach. In this section, we formulate a framework to estimate the expected computational complexity for the branch-and-bound model selection approach presented in the previous section.

Under the current formulation, each node of the solution tree represents two hypotheses, the optimal solution hypothesis and the partial solution hypothesis. This

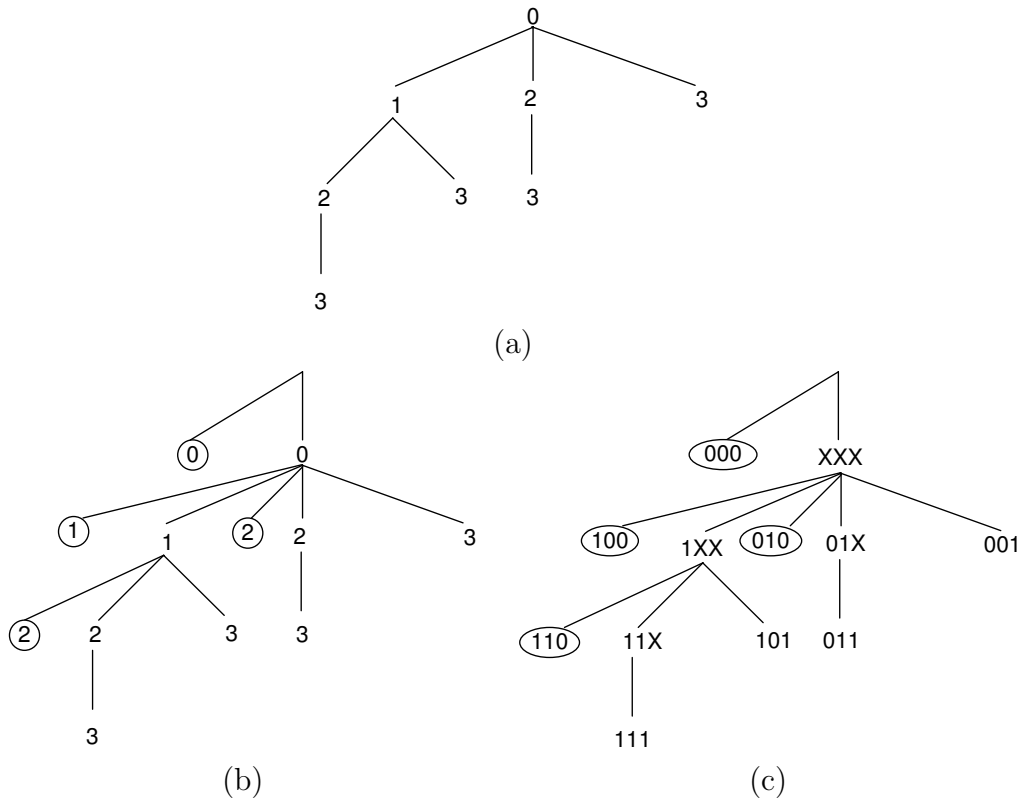


Figure 4.2. (a) Original branch-and-bound tree for $N_c = 3$, (b) its edge-weighted equivalent, (c) Coding for the tree nodes.

gives rise to a binomial tree [65] of order N_c as the representation of the model selection problem (see figure 4.2(a)). However, in a typical tree search problem only the leaf nodes can represent an optimal solution. To incorporate this, we modify the original tree structure and add a "twin" node to each internal node of the binomial tree. This updated tree structure is shown in figure 4.2(b) and will be used for computational complexity analysis. The circled nodes are the newly added twin nodes. In the updated tree, the leaf nodes (i.e., leaf nodes from the original tree and newly added twin nodes) represent the optimal solution hypotheses and the internal nodes represent the partial solution hypotheses.

To represent each hypothesis uniquely, we devise a representation for each hypothesis with symbols $\{0, 1, \mathsf{X}\}$ ("zero", "one" and "undetermined"). In this N_c elements wide representation, if a hypothesis includes a candidate C_i then $(N_c - i)$th element of the representation is '1' and if the hypothesis does not include the candidate C_i then $(N_c - i)$th element is '0'. For the partial solution hypothesis represented by internal nodes, an additional symbol 'X' is used. The symbol X indicates a candidate that can be included in a solution later in the search. The $(N_c - i)$th element of the representation is set to X, if any child nodes can include the candidate C_i. One can quickly get the "twin" node of an internal node by replacing 'X's with '0's.

The cost associated with a leaf node is the cost of the optimal solution hypothesis $\mathcal{C}(\Theta)$. On the other hand, the cost associated with an internal node is the cost of the partial solution hypothesis $\mathcal{C}_{\text{Lower}}(\Theta)$. From (4.11) and (4.13), one can conclude that the cost of a node is lower than its child nodes. This also means that each edge of the updated tree has a nonzero positive weight associated with it. The cost of reaching a node can be computed by adding weights of all the edges along the path from the node to the root of the tree. Note that, our formulation does not require explicit computation of the edge weights as the cost of reaching a node can be directly computed from (4.11) or (4.13). The least cost leaf node in the edge-weighted tree corresponds to the optimal solution for the branch-and-bound process. Thus, our branch-and-bound approach can be seen as a least cost leaf search problem for the updated edge-weighted tree.

4.3.1 Average Complexity

To estimate the average complexity, we concentrate on the BBF approach, which explores the least number of nodes before it reaches the optimal solution [86].

In a BBF implementation, every time the least cost node is popped out of the priority queue Q. Child nodes of the currently popped node are inserted in the queue Q. The priory of a node is set inversely proportional to its cost. For the edge weighted tree, the first leaf node popped from Q during BBF search is optimal [86]. This also means that the complexity, i.e. the number of nodes popped out before the optimal node, is same as the number of internal nodes with costs less than the optimal cost. Additionally, the optimal node has the cost less than all the other leaf nodes by definition.

Let T denote the set of all the leaf nodes of the tree and I denote the set of all internal nodes of the tree. The optimality probability of the node i, $\Pr_o(i)$, denotes the probability that the node i is optimal, i.e. it has the least cost among the leaf nodes.

$$\Pr_o(i) = \prod_{\forall j \in T \setminus i} \Pr(\mathcal{C}(i) < \mathcal{C}(j)) \tag{4.18}$$

The cost probabilities $\Pr(\mathcal{C}(i) < \mathcal{C}(j))$ are probabilities of comparison of the sum of edge weights leading to nodes i and j. These can be seen as probabilities of comparison between two sums of edge weights and thus can be given by $\Pr_T(S_m < S_n)$. Here S_m and S_n are the sums of m and n edge weights respectively ($1 \leq n \leq N_c, 1 \leq m \leq N_c$). Note that it is not necessarily true that $m = \mathit{depth}(i)$ and $n = \mathit{depth}(j)$. One has to remove the common edges along the path to the root node from the node depth to get values of m and n. If the number of common edges is l then $m = \mathit{depth}(i) - l$ and $n = \mathit{depth}(j) - l$. We define nodes i and j to have a relationship of order (m, n). In graph theory terms, the relationship between the two nodes can be seen as the simple path between them and $(m + n)$ gives the length of the simple path.

Due to the recursive structure of the tree, the weight relationships repeat themselves. Thus, $\Pr_o(i)$ can be written as,

$$\Pr_o(i) = \prod_{m=1}^{N_c} \prod_{n=1}^{N_c} \Pr_T(S_m < S_n)^{O_i(m,n)} \tag{4.19}$$

Here O_i is the optimality matrix for the node i and its (m,n)th element indicates the number of times the relationship (m,n) (and hence the term $\Pr_T(S_m < S_n)$) appears in the computation of $\Pr_o(i)$.

The complexity for node i, $N(i)$ denotes the number of internal nodes explored by BBF search if the node i is optimal. When the node i is optimal, the internal node j is explored only if its cost is less than the cost of the optimal node i. Thus, the complexity when the node i is optimal is,

$$N(i) = \sum_{\forall j \in I} \Pr(\mathcal{C}_{\text{Lower}}(j) < \mathcal{C}(i)) \tag{4.20}$$

Similar to the optimality probability $\Pr_o(i)$, the complexity $N(i)$ of the node can be expressed as,

$$N(i) = \sum_{m=1}^{N_c} \sum_{n=1}^{N_c} \Pr_I(S_n < S_m) \cdot R_i(m,n) \tag{4.21}$$

Here R_i is the complexity matrix for the node i and its (m,n)th element indicates the number of times the relationship (m,n) (and hence the term $\Pr_I(S_m < S_n)$) repeats in computation of $N(i)$. Note that different subscripts are used for probabilities $\Pr_T$ and $\Pr_I$, as the sums compared by these probabilities differ slightly. For $\Pr_T$, one of the weights in both sums is for an edge from an internal node to a leaf node while all the other weights are for edges between internal nodes. For $\Pr_I$, all the weights

correspond to edges between internal nodes. If we assume that this difference is negligible then,

$$\Pr(S_m < S_n) = \Pr_T(S_m < S_n) = 1 - \Pr_I(S_n < S_m). \tag{4.22}$$

With the optimality probability $\Pr_o(i)$ and the complexity $N(i)$, the expected complexity $\overline{N}$ can be estimated as,

$$\overline{N} = \frac{\sum_{\forall i \in T} \Pr_o(i) N(i)}{\sum_{\forall i \in T} \Pr_o(i)}. \tag{4.23}$$

The following section describes the computation of quantities involved in the estimation of the expected complexity.

4.4 Cost Probabilities, Optimality and Complexity Matrices

The optimality matrix O_i and the complexity matrix R_i are different for nodes that do not have $(1, 1)$ relationship and, the matrices change with the order of the tree as well. However, the cost probabilities $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ are only determined by the distribution of edge weights. This section first describes how to estimate these probabilities.

4.4.1 Cost Probabilities for Uniformly Distributed Edge Weights

This section estimates cost probabilities for an edge weighted tree with edge weights uniformly distributed between $[0, 1]$. We start with the sum of m independently and uniformly distributed edge weights between $[0, 1]$ given by [93],

$$f_m(S_m) = \frac{1}{(m-1)!} \sum_{j=0}^{m} (-1)^j \binom{m}{j} [(S_m - j)_+]^{m-1} \tag{4.24}$$

Here, $(\cdot)_+$ means positive part of $(\cdot)$. This can be written as,

$$(\cdot)_+ = \frac{(\cdot) + |(\cdot)|}{2}$$

$\Pr(S_m < S_n)$ when $m > n$ can be derived from $f_m(S_m)$ to be,

$$\begin{aligned} \Pr(S_m < S_n) &= \sum_{q=1}^{n} \sum_{k=0}^{q-1} \sum_{j=0}^{q-1} (-1)^{(k+j)} \binom{n}{k} \binom{m}{j} \\ &\quad \left[\sum_{p=1}^{n} (-1)^{(p-1)} \frac{(x-k)^{(n-p)}}{(n-p)!} \frac{(x-j)^{(m+p)}}{(m+p)!} \right]_{(q-1)}^{q} \end{aligned} \tag{4.25}$$

Here,

$$[f(x)]_{(q-1)}^{q} = f(q) - f(q-1)$$

Refer to the appendix for the detailed derivation. $\Pr(S_m < S_n)$ when $m < n$ can simply computed as,

$$\Pr(S_m < S_n) = 1 - \Pr(S_n < S_m).$$

Figure 4.3 shows the plot of these probabilities.

4.4.2 Cost Probabilities by Sampling

For a typical model selection problem, the distribution for the sums does not have a closed form solution or it is unknown. In such a case, a close approximation of $\Pr_T(S_n < S_m)$ and $\Pr_I(S_n < S_m)$ or $\Pr(S_n < S_m)$ can be generated with sampling. The sampling can be implemented as a simple process listed below.

- For all the possible combinations of m and n repeat following N_s times.
  - Generate a hypothesis Θ_1 of size m and compute its cost $\mathcal{C}(\Theta_1)$.

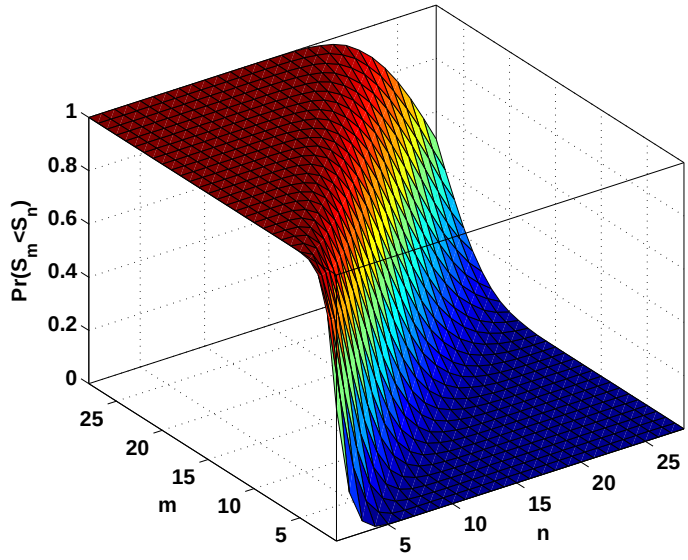


Figure 4.3. $\Pr(S_m < S_n)$ for uniform iid random variables.

  - Generate a hypothesis Θ_2 of size n such that $\Theta_1 \cap \Theta_2 = \emptyset$ and compute its cost $\mathcal{C}(\Theta_2)$.
  - Compare the costs of the hypotheses, if $\mathcal{C}(\Theta_1) < \mathcal{C}(\Theta_1)$, $\Pr(S_m < S_n) = \Pr(S_m < S_n) + 1$.
- For all the possible combinations of m and n, normalize the probabilities, $\Pr(S_m < S_n) = \Pr(S_m < S_n)/N_s$.

Figure 4.4 shows $\Pr(S_m < S_n)$ generated with sampling when the edge weights are independent and identically distributed (iid) as squared zero mean Gaussian with unit standard deviation respectively.

4.4.3 Computing Optimality Matrix

The optimality matrix O_i is computed by comparing each leaf node i with all the other leaf nodes. The edge weighted tree can be seen as a binomial tree with an

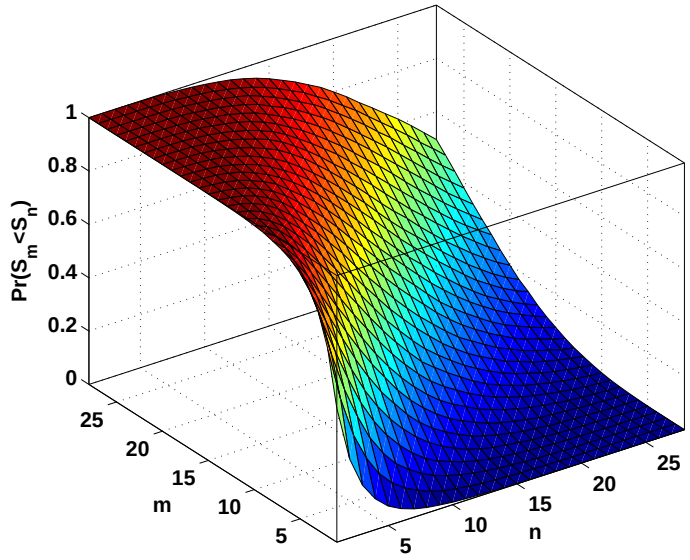


Figure 4.4. $\Pr(S_m < S_n)$ for squared Gaussian iid random variables generated by sampling.

added "twin" node for all the internal nodes. Thus, the recursive properties of the binomial tree can be used in the computation of O_i. To compute O_i, we transform the edge weighted tree back to the binomial tree by merging the twin nodes with the internal nodes and retaining the representation of the twin nodes after merging. Thus, the node representation is now binary.

Computation of O_i relies on the property that *a simple path between two nodes of a binomial tree includes the root node of only the smallest subtree including both the nodes*. Note the following important properties before proceeding to compute O_i.

- Depth of a node, $d(i)$ is equal to the number of ones in the binary representation.
- Each node i belongs to a unique combination of binomial subtrees

$$T_0, T_1, \ldots, T_{d(i)}$$

and the location of ones in the representation indicates the order of binomial subtrees, e.g. the most significant bit indicates a binary subtree of order $N_c - 1$ and the least significant bit indicates a binomial subtree of order 0. Note that all the nodes belong to a subtree T_0 of order N_c.

- The number of nodes belonging to a subtree T_t at depth k is given by,

$$N_t(k) = \begin{cases} \binom{T_t}{k-t}, & \text{if } 0 \leq k - t \leq T_t; \\ 0, & \text{otherwise.} \end{cases} \tag{4.26}$$

where $k = 0, 1, 2, \ldots, N_c$ and $t = 0, 1, 2, \ldots, d(i)$.

Since a node i only belongs to subtrees $T_0, T_1, \ldots, T_{d(i)}$, to compute O_i, we have to analyze these subtrees alone. The binomial tree can be split into these subtrees and can be analyzed subtree by subtree, starting with the largest subtree T_0. For each subtree, we select the nodes that exclusively belong to the subtree under consideration. This can be accomplished by removing the nodes belonging to the next largest subtree from the subtree under consideration. Finally, one has to offset the result of merging of the "twin" nodes. The merging leads to the relationships of the order $(m, 0)$ and $(0, n)$, which would have been of the order $(m + 1, 1)$ and $(1, n + 1)$ otherwise. Also, we have to remove the relationship $(0, 0)$, which corresponds to a comparison of the node i with itself. The algorithm to compute the optimality matrix O_i follows.

1. Initialize $T = \{T_0, T_1, \ldots, T_d\}$ = the subtree membership of the node i, $d(i)$ = depth of the node i, set $O_i(1, 1) = -1$ and all the other elements of O_i equal to zero.
2. Set $t = 0$ such that the current subtree $T_t = T_0$.

3. For the subtree T_t, at each depth $k = 0, 1, \ldots, N_c$ compute $M_t(k)$ the number of nodes, which belong exclusively to the subtree tree T_t.

$$M_t(k) = \begin{cases} N_t(k) - N_{t+1}(k), & \text{if } t < d(i); \\ N_t(k), & \text{Otherwise.} \end{cases}$$

4. If $k > 0$ and $d(i) - t > 0$, set $O_i(d(i) - t, k) = O_i(d(i) - t, k) + M_t(k)$ else set $O_i(d(i) - t + 1, k + 1) = O_i(d(i) - t + 1, k + 1) + M_t(k)$.
5. Set $t = t + 1$. If $t \leq d(i)$, then go to step (3), else terminate the algorithm.

4.4.4 Computing Complexity Matrix

To compute the complexity matrix R_i, each leaf node i has to be compared with internal nodes of the cost weighted tree. After the "twin" node merging, one has to compare each node i of the merged tree with all the internal nodes of the tree. Note that the internal nodes of a binomial tree of order N_c form a binomial tree of order $(N_c - 1)$. Thus, similar to (4.26) the number of internal nodes belonging to subtree T_t at depth k is given by,

$$L_t(k) = \begin{cases} \binom{T_t - 1}{k - t}, & \text{if } 0 \leq k - t \leq T_t - 1; \\ 0, & \text{otherwise.} \end{cases} \tag{4.27}$$

The algorithm to compute the complexity matrix R_i is a slight variation of the one that calculates O_i.

1. Initialize $T = \{T_0, T_1, \ldots, T_d\}$ = the subtree membership of the node i, $d(i)$ = depth of the node i, set all the other elements of R_i equal to zero.
2. Set $t = 0$ such that the current subtree $T_t = T_0$.

3. For the subtree T_t, at each depth $k = 0, 1, \ldots, N_c$ compute $M_t(k)$ the number of nodes, which belong exclusively to the subtree tree T_t.

$$M_t(k) = \begin{cases} L_t(k) - L_{t+1}(k), & \text{if } t < d(i); \\ L_t(k), & \text{Otherwise.} \end{cases}$$

4. If $m > 0$ and $d(i) - t > 0$, set $R_i(d(i) - t, k) = R_i(d(i) - t, m) + M_t(k)$ else set $R_i(d(i) - t + 1, k + 1) = R_i(d(i) - t + 1, k + 1) + M_t(k)$.
5. Set $t = t + 1$. If $t \leq d(i)$, then go to step (3), else terminate the algorithm.

Note that when the internal node j is an ancestor of the leaf node i, the probability $\Pr(S_n < S_m)$ is 1. We have to correct the complexity matrix for these cases by setting $R_i(k, 1) = R_i(k, 1) - 1$ where $0 \leq k < d(i)$. With the corrected complexity matrix R_i, the complexity $N(i)$ becomes,

$$N(i) = d(i) + \sum_{m=1}^{N_c} \sum_{n=1}^{N_c} \Pr(S_n < S_m) \cdot R_i(m, n) \tag{4.28}$$

Since the complexity for computing the optimality matrix and the complexity matrix increases exponentially with N_c, the computational complexity analysis for the model selection can only be accomplished for a moderate number of candidates. In the following section, computational complexity of branch-and-bound model selection is analyzed for the multiple structure-and-motion segmentation.

4.5 Experimental Results

We studied the performance of the proposed model selection framework for multiple structure-and-motion (MSaM) segmentation problem. The MSaM segmentation is carried out on a pair of images. We chose publicly available image data

sets for our experiments. To generate candidates for the model selection, for each image correspondence, a fundamental matrix candidate was computed from a circular spatial neighborhood of the correspondences using the "Structure-and-Motion Toolkit" from [67]. Outlier and inlier correspondences were selected for each fundamental matrix candidate by applying a threshold δ_T to the reprojection error of the correspondences for the candidate. The number of inlier correspondences for each candidate indicates the support for the candidate. To avoid repeated candidates, which are similar, candidates with smaller support sharing substantial ($> 80\%$) inlier correspondences with a candidate with larger support, were suppressed. Finally, the surviving candidates were arranged in decreasing order of their support. The Bayesian information criterion (BIC) was optimized for these candidates to select the optimal hypothesis. Some correspondence are tagged as outliers as none of the selected candidates can explain them with residuals less than δ_T.

The complexity of proposed branch-and-bound model selection approach was estimated with synthetic data. For the experiments, 100 different fundamental matrices were randomly generated. For each fundamental matrix, 50 correspondences were generated with the model given by (4.15) adding iid Gaussian noise with $\sigma = 1$. At a time, correspondences for four different fundamental matrices were combined together to form an experimental data set. Additionally, 50 randomly selected correspondences from the remaining motions were added to the data as outliers. The number of hypotheses N_c cannot be explicitly controlled and it varies with the number of motions and their spatial configuration. For our synthetic data, N_c was close to 20 to 30. To estimate the probabilities $\Pr_T(S_m < S_n)$ for the MSaM segmentation problem, we randomly generated pairs of hypotheses and compared their BIC values. To calculate the probabilities $\Pr_I(S_m < S_n)$, BIC value of a randomly generated hypothesis was compared to the lower bound on BIC value of another randomly gen-

erated hypothesis. Figures 4.5 and 4.6 show the probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ respectively.

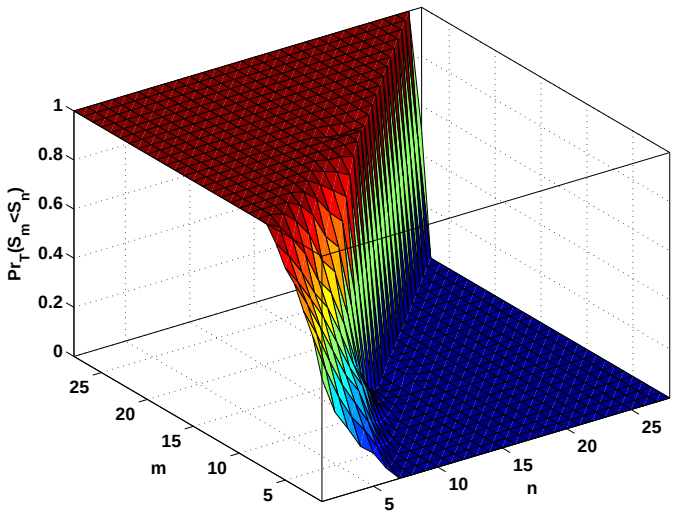


Figure 4.5. $\Pr_T(S_m < S_n)$ for the MSaM segmentation problem.

Once the probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ are known from sampling, the complexity for the branch-and-bound search can be estimated by evaluating (4.23). Figure 4.7 shows the estimated expected complexity for the MSaM segmentation problem along with other tree search problems when edge weights are uniform iids and squared Gaussian iids. The worst case complexity, which is equivalent to a brute force search, is also shown for comparison. Clearly, the expected complexity of the branch-and-bound search depends on the distribution of the edge weights. This distribution is captured by probabilities $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$.

As seen from the plots, although the expected complexity is much lesser than the worst case complexity for the branch-and-bound, it remains exponential for the

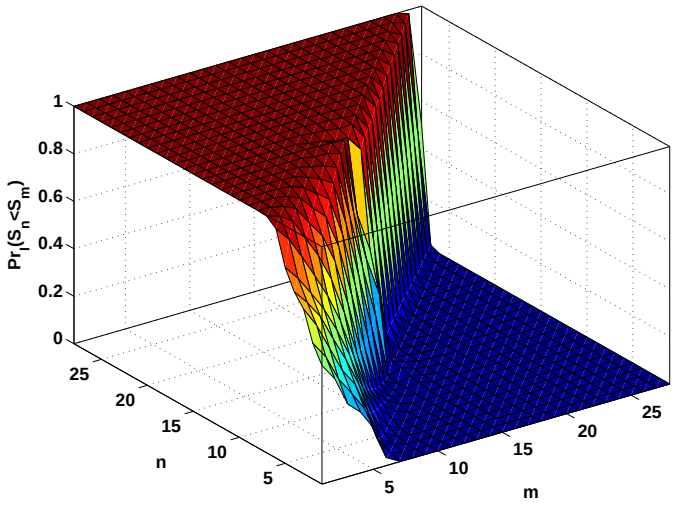


Figure 4.6. $\Pr_I(S_n < S_m)$ for the MSaM segmentation problem.

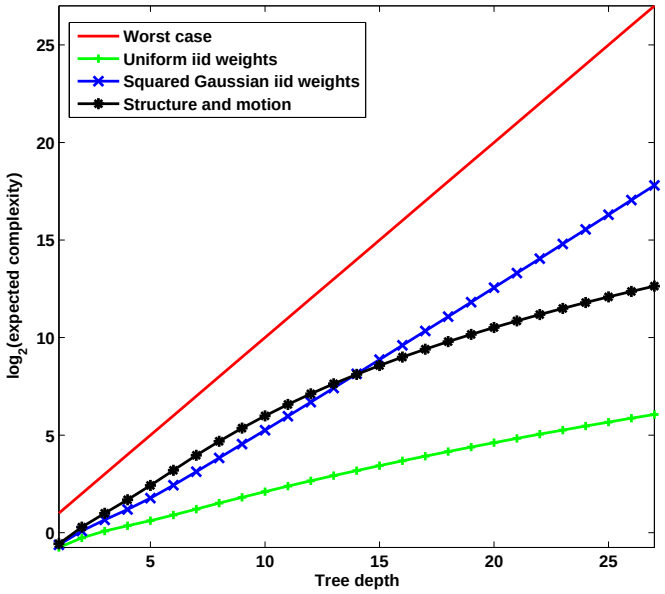


Figure 4.7. Expected complexity.

most part. The rate of exponential depends on how quickly the off diagonal values of probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$ drop to near zero/ rise close to one. On the other hand, for the MSaM segmentation problem, the increase in the complexity as $N_c > 15$ is not as drastic as $N_c < 15$. This again is a result of the off diagonal values of probability matrices $\Pr_T(S_m < S_n)$ and $\Pr_I(S_n < S_m)$, almost all of which drop to near zero/ rise close to one for $N_c > 15$.

Figure 4.8 compares the estimated expected complexity of the problem with the experimentally observed complexity of the problem. We ran 400 experiments with different data sets to find the number of nodes explored before optimal solution was found. These experiments were then separated based on value N_c and sorted according to increasing complexity. The lengths of plots were normalized horizontally to 100 for easy comparison of complexity for various values of N_c. As seen in figure 4.8, although the expected complexity is slightly overestimated, it still provides a satisfactory estimate for the observed complexity.

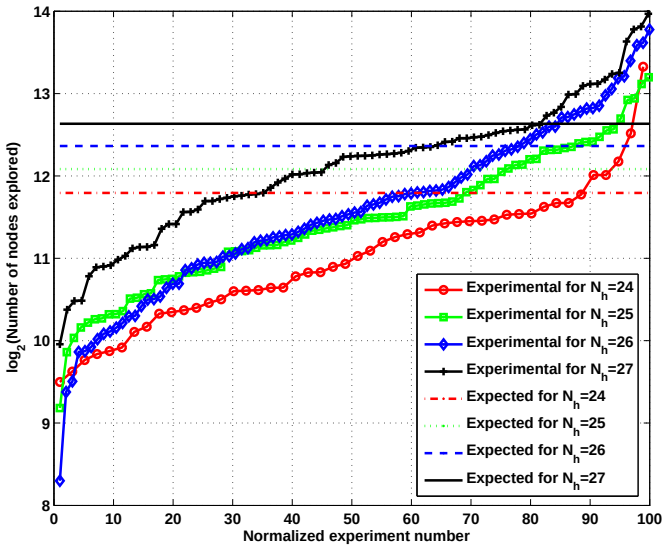


Figure 4.8. Comparison of expected and actual complexity.

After proposing branch-and-bound based segmentation methods in last two chapters and, establishing computational complexity of the branch-and-bound, the following chapter concentrates on the problem of visual object recognition.

CHAPTER 5

VISUAL OBJECT RECOGNITION

5.1 Introduction

The visual bag-of-words approach is the predominantly applied approach to solve the visual object recognition problem [94, 95]. In the visual bag-of-words approach, each image is characterized by a histogram of the quantized image features. Each image feature describes an invariant local interest point [96] extracted with detectors such as Harris-affine interest point detector [97], difference-of-gaussian detector [98], maximally stable extremal regions [99] etc. Each image feature describes an image patch around the detected interest point. The descriptions of the interest points are high dimensional vectors. Mikolajczyk and Schmid [100] compare various descriptors available in the literature and conclude that scale invariant feature transform (SIFT) [98] is the most effective description for the image patches. To reduce dimensionality of the image descriptors, vector quantization has to be is applied to the features before a histogram can be constructed. A clustering algorithm such as K-means or hierarchical K-means is applied to the image features and, cluster centers are established. The image features are quantized by assigning them to the closest cluster center. Normally, one histogram per image is formed for the quantized image features. In the final step of the visual bag-of-words classifier, histograms for training images are used to train a classifier with a machine learning algorithm such as support vector machine (SVM), boosting or neural networks.

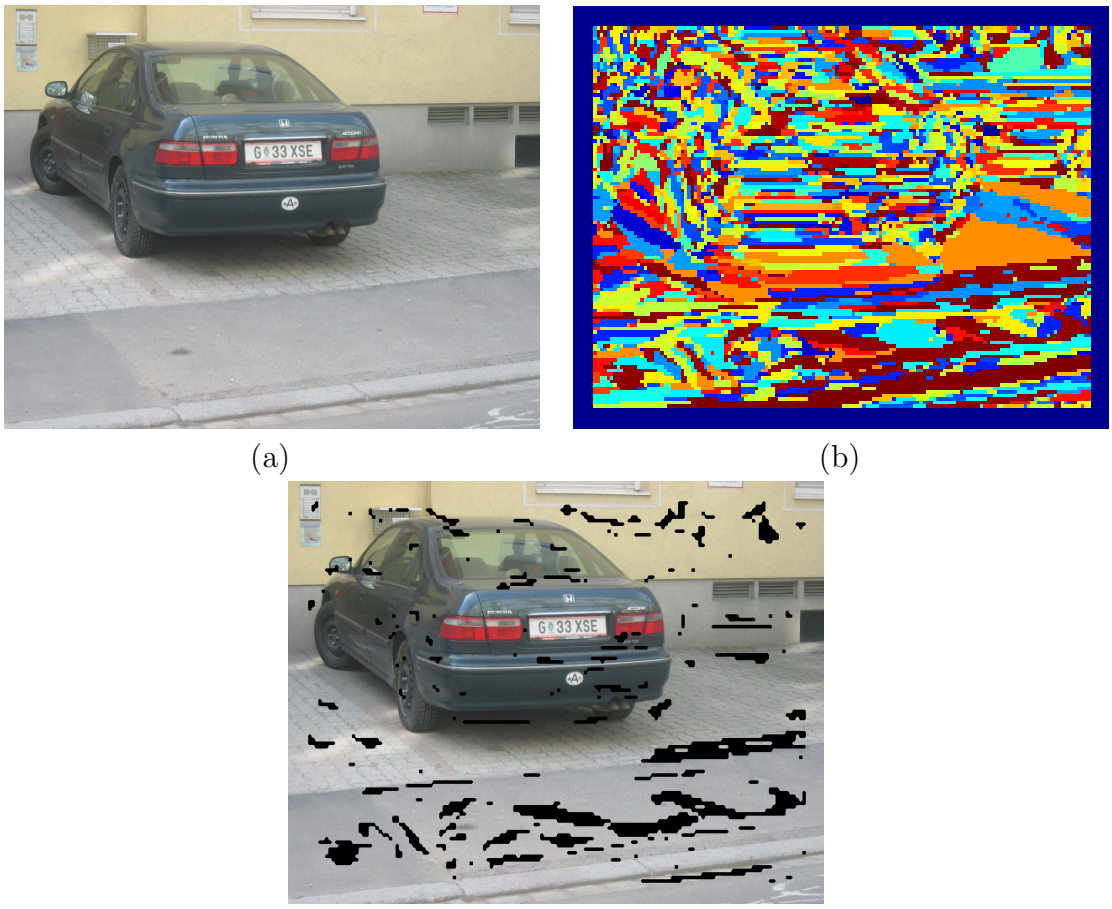
For an image classification problem, the image patches which appear frequently in positive examples can be seen to have a positive relevance to the recognition and,

the ones which appear frequently in negative examples can be seen to have a negative relevance. Note that the image patches referred here can be larger than the patch described by an image feature. In the histogram, an image feature carries equal weight regardless of whether it appears as a part of a positively or negatively relevant image patch, as in both cases the quantization results in the same representation. Intuitively, the accuracy of the classification should be improved if positive and negative relevance of an image location is estimated and the image features are weighted accordingly to form two separate histograms. We propose a two-step framework, which first estimates positive and negative relevance of an image location. The second step of the framework uses the relevance of the image location estimated in the first step to weight the image features and classifies the object with the conventional visual bag-of-words approach applied to the weighted histograms.

This chapter is organized as follows: Section 5.2 motivates how a sliding window SVM classifier can be used in the proposed relevance framework. Approach to estimate the relevance is described in section 5.3. Relevance weighting applied in the bag-of-features classifier is explained in section 5.4. Experimental results are presented in section 5.5.

5.2 Motivation

Figures 5.1(a) and (b) show an image and its quantized equivalent respectively. As seen from 5.1(c), the same quantized feature appears on the car as well as the background. While forming a histogram to represent the image, all the marked features will be added to a single histogram bin. Our goal is to establish a framework, which treats an image feature based on its location, while maintaining simplicity of the bag-of-features approach.



(a) (b)

(c)

Figure 5.1. (a) Original image, (b) Quantized features, (c) Original image overlaid with locations of features from one of the histogram bins marked as black dots.

Consider an image with M dense features

$$\{f_1, f_2, f_3, \ldots, f_M\}$$

located at

$$\{l_1, l_2, l_3 \ldots, l_M\}.$$

After quantization the features to K levels, they belong to histogram bins

$$\{b_1, b_2, b_3, \ldots, b_M\}$$

respectively, where $b_m \in \{1, 2, \ldots, K\}$.

Support vector machines (SVMs) are frequently used as classifiers in the visual bag-of-words classification approach. It can be easily shown that the linear kernel assigns a weight to each feature based on its histogram bin. The decision function for an SVM is given by,

$$y(x) = \beta + \sum_{i=1}^{N} \alpha_i k(x, x_i) \tag{5.1}$$

where $k(\cdot, \cdot)$ represents the kernel function, x_i is one of the N support vectors and α_i is the corresponding weight learned. In case of a visual bag-of-words classification approach, the observations to be classified x are histograms, which can be normalized or non-normalized. A non-normalized histogram can be denoted by H and a normalized histogram by $\hat{H}$. Rewriting (5.1) for visual bag-of-words representation with a linear kernel

$$y(\hat{H}) = \beta + \sum_{i=1}^{N} \alpha_i \langle \hat{H}, \hat{H}_i \rangle \tag{5.2}$$

where $\langle \cdot, \cdot \rangle$ represents the inner product. If the histogram has K bins and kth bin has h^k features, the kth normalized bin entry can be given by

$$\hat{h}^k = \frac{h^k}{\sum_{k'=1}^{K} h^{k'}} = \frac{h^k}{M}.$$

A partial histogram for an mth feature can be represented by $\tilde{H}^m$, which has only b_mth histogram entry equal to one and, all the other entries equal to zero. A nor-

malized histogram can be built from the partial histograms of the individual features as,

$$\hat{H} = \frac{1}{M}\sum_{m=1}^{M}\tilde{H}^m \tag{5.3}$$

Inserting (5.3) in (5.2),

$$\begin{aligned} y(\hat{H}) &= \beta + \sum_{i=1}^{N}\alpha_i\langle\frac{1}{M}\sum_{m=1}^{M}\tilde{H}^m, \hat{H}_i\rangle \\ &= \beta + \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N}\alpha_i\langle\tilde{H}^m, \hat{H}_i\rangle. \end{aligned}$$

After expanding the inner product $\langle\tilde{H}^m, \hat{H}_i\rangle$ to a summation of product, only one of the product terms is nonzero. This leads to,

$$\begin{aligned} y(\hat{H}) &= \beta + \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N}\alpha_i h_i^{b_m} \\ &= \beta + \frac{1}{M}\sum_{m=1}^{M} w^{b_m}. \end{aligned}$$

Here, the weight of feature belonging to kth histogram bin is given by

$$w^k = \sum_{i=1}^{N}\alpha_i h_i^k.$$

A feature, which appears frequently in the positive support vectors, would have a positive weight and, a feature appearing in the negative support vectors would have a negative weight.

The weights established for the feature f_m can be possibly used as relevance for its location l_m. Intuitively, positive or negative relevance of an image location in the classification would mean that its neighborhood is also relevant in a similar way. As the SVM training does not involve spatial information, the weights given

above are not smoothly distributed over the neighborhood. To force smoothness on the relevance over the neighborhood, one can average the above weights over a local window.

After averaging weights over a local window, the process reduces to evaluating the SVM decision function over the local window. Thus, the relevance at an image location l_m can be established by evaluating the SVM classifier over a local window N_{l_m}. Although, the discussion here was limited to the linear kernel, from its conclusion, a similar sliding window implementation for relevance of location should be possible for non-linear kernels.

5.3 Estimating Relevance

The localization approach by Fulkerson *et al.* [101], who apply brute force sliding window to localize an object in an image, is adapted to estimate the positive and negative relevance of the image patches. As the relevance has to be established for the entire image, dense SIFT features are extracted from the image. To generate a quantized representation of the features, hierarchical K-means clustering is applied to the extracted dense features. To speed up the SVM classifier, which has to be applied repeatedly, the quantized representations of the features is compressed with agglomerative information bottleneck (AIB). The compression allows representation of the image with a few hundred histogram bins instead of thousands of bins. The compressed histograms are used to train an SVM classifier. To establish the positive and negative relevance of an image location l_m, the SVM classifier is applied to a local window N_{l_m} around the location to generate a raw decision value with (5.1). As the raw SVM outputs cannot be used directly to weight the features, probabilistic SVM outputs are generated by fitting a parametric sigmoid to the raw SVM outputs during training as suggested by [102]. The estimated posterior probabilities are used

as the relevance weights to form soft histograms in the second step of the classifier. The positive relevance weight at location l_m is given by,

$$W^{+}(l_m) = \Pr\left(+\text{ve Class}|y(N_{l_m})\right). \tag{5.4}$$

Similarly, the negative relevance weight is,

$$W^{-}(l_m) = \Pr\left(-\text{ve Class}|y(N_{l_m})\right) = 1 - W^{+}(l_m) \tag{5.5}$$

5.4 Relevance Weighted Bag-of-Features Classifier

In the second step of the classifier, two separate soft histograms are created: one corresponding to features belonging to positively relevant image areas and the other corresponding to features belonging to negatively relevant area. The soft histograms corresponding to positively and negatively relevant image patches are given by,

$$\left(h^{k'}\right)^{+} = \sum_{\forall n, b_n = k'} W^{+}(l_n) \tag{5.6}$$

$$\left(h^{k'}\right)^{-} = \sum_{\forall n, b_n = k'} W^{-}(l_n) \tag{5.7}$$

Note that, the different index n indicates that the features in the second step can be different and sparse in the second step. Use of k' suggests that the features can be quantized differently compared to the first step. Thus, for the second step of the classifier, any variant of the bag-of-features classifier that uses histogram representation of the image can be applied. The only modification required is substitution of the original histogram representation with appended soft histograms.

Figure 5.2 summarizes the two step relevance weighted visual bag-of-features approach.

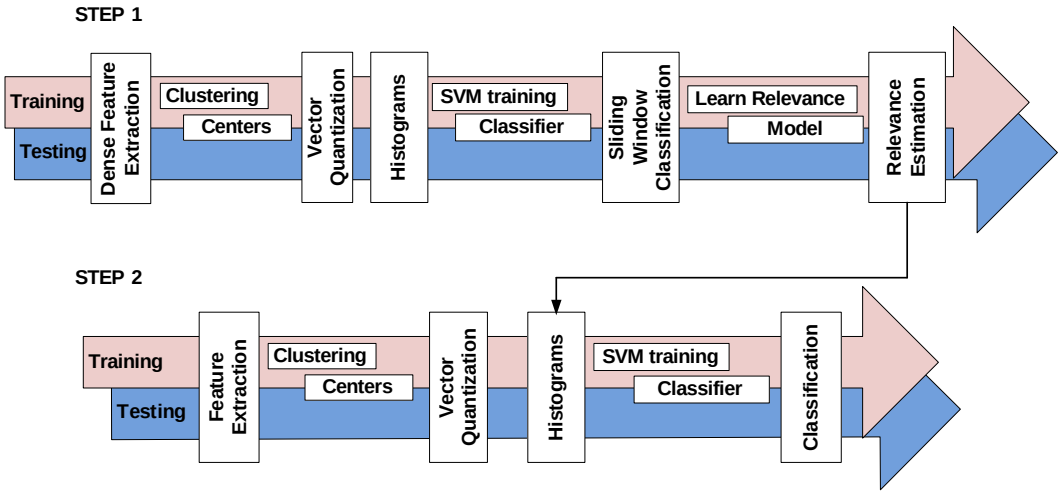


Figure 5.2. Relevance weighted visual bag-of-features approach.

5.5 Experimental Results

The proposed relevance based two-step classifier was validated with the challenging *Graz02* data set [103]. *Graz02* data set is split into four classes: bike, car, person and background. For each class, the data set provides more that 300 example color images. For each of the object classes, a binary classifier is designed to classify the object against the background class. For the training, the first 150 odd numbered images from each class are used, while for the testing, the first 150 even numbered images are used. To implement the proposed framework, we used "VLFeat" library [104] and "Blocks" modular framework [105].

For each image, color dense SIFT features were extracted for every 4th pixel of the image. This leads to about 19,000 features per image. Each feature characterizes a 16×16 spatial window of the image. Hierarchical K-means with $K = 10$ with $10,000$

leaf nodes was carried out to construct an initial dictionary, which was compressed to 40 visual words with AIB. Based on the visual words, the dense features were quantized and histograms were constructed from the training images. An SVM classifier with chi-square kernel was trained with LIBSVM [106]. The parameters of SVM and the kernel were tuned with cross validation. LIBSVM was configured to generate the posterior probability estimates, which were used as relevance weights in the second step of classifier. To estimate the relevance of an image location, the SVM classifier was applied to a square window of size 80×80 centered at the location. For the points where the relevance estimate was unavailable, the estimate was generated by interpolation.

For the bag-of-words classifier in the second step, conventional sparse SIFT features were used. Similar to the first stage, clustering was carried out and a compressed dictionary with 1000 visual words was constructed. However, instead of creating a single histogram after quantization, two soft histograms were generated and appended together for training. An SVM classifier with chi-square kernel was designed with the appended histograms. A single histogram based SVM was also trained for comparison with the conventional approach.

The performance of the proposed classification scheme is compared with the conventional bag-of-words classifier with two measures: the equal error rate and area under the receiver operating characteristic (ROC) curve. Equal error rate is achieved when the classification accuracies of the positive class and the negative class are equal. In the ROC curves shown in figure 5.3, this point is indicated by the intersection of ROC with the diagonal equal error line. Additionally, it is desirable that the area under ROC is close to 1.

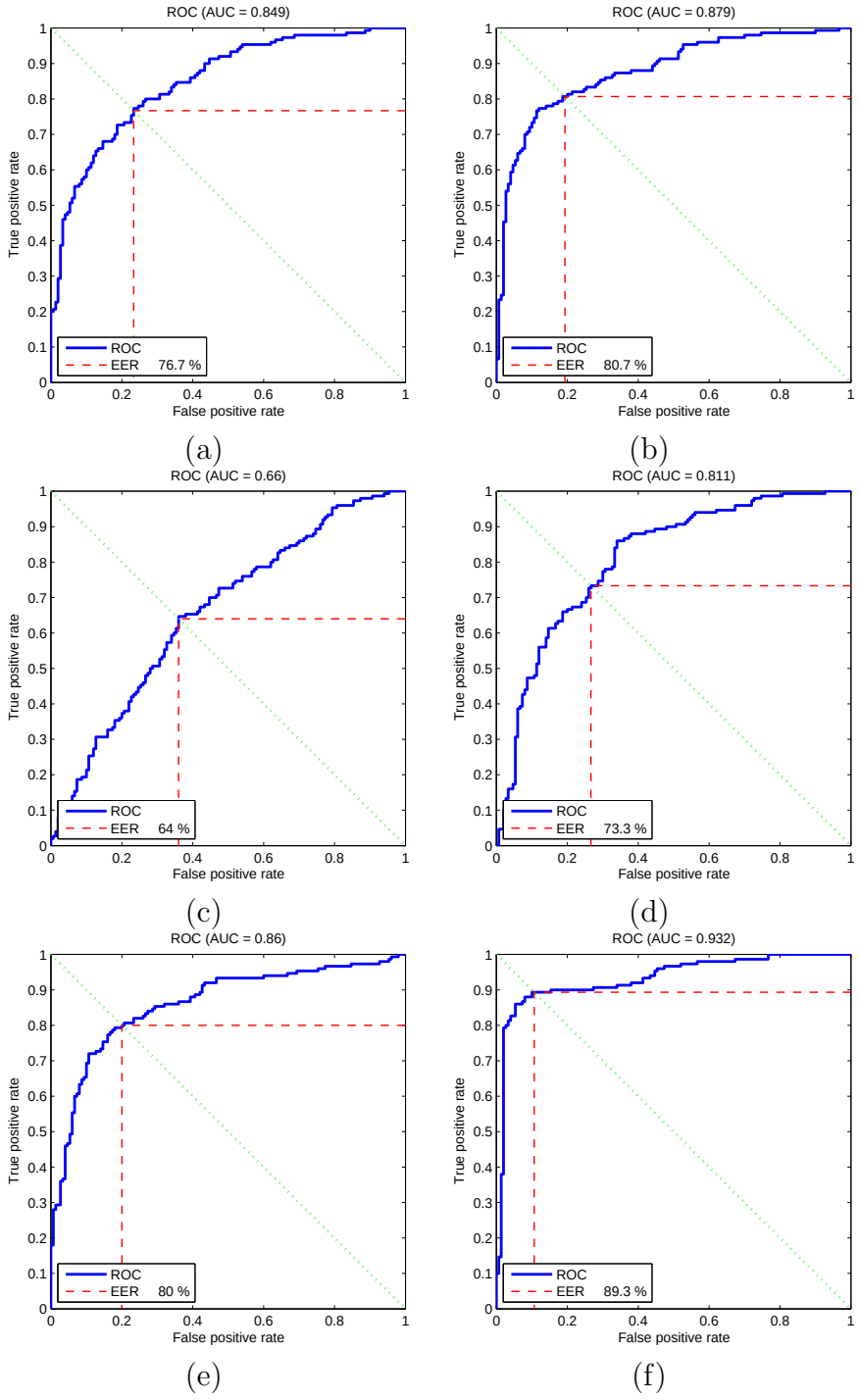


Figure 5.3. Graz02: ROC curves Left: Without relevance separation Right: With relevance separation for (a)(b) Class "Bike,"(c)(d) Class "Cars," (e)(f) Class "Person".

As seen from the tables 5.1 and 5.2, the proposed two step method outperforms the conventional bag-of-features method for all the classes. The complete ROCs for all the classes can be seen in figure 5.3.

Table 5.1. Graz02: Chi square kernel with 40 visual words used for relevance estimation: Equal error rate in % averaged over 10 runs

Class	Without relevance	With relevance
Bike	76.73	81.26
Car	61.2	73.86
Person	79.86	87.20

Table 5.2. Graz02: Chi square kernel with 40 visual words used for relevance estimation : Area under ROC curve averaged over 10 runs

Class	Without relevance	With relevance
Bike	0.85	0.89
Car	0.66	0.81
Person	0.86	0.92

In figure 5.4, some of the test images are shown, for which the proposed classifier labels them correctly at least eight out of ten times while the conventional classifier fails to classify them correctly nine or ten out of ten times. Looking at the corresponding positive relevance weights, it can be concluded that the relevance weights are assigned according to location of the object. Although, a significant portion of the background is still marked with positive relevance, the proposed classifier succeeds in classifying the image.

On the other hand, figure 5.5 shows images, for which the proposed classifier fails at least nine out of ten times while the conventional classifier fails at most 2

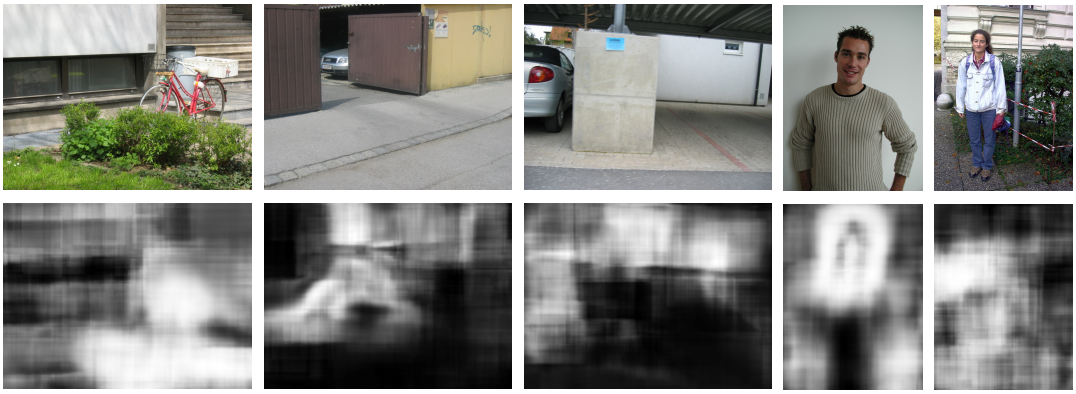


Figure 5.4. Some of the test images and the corresponding positive relevance weights, for which the relevance weighted classification significantly outperforms the conventional classifier.

times. Failure of the proposed classifier can be clearly attributed to the incorrect relevance weights estimated for the objects.

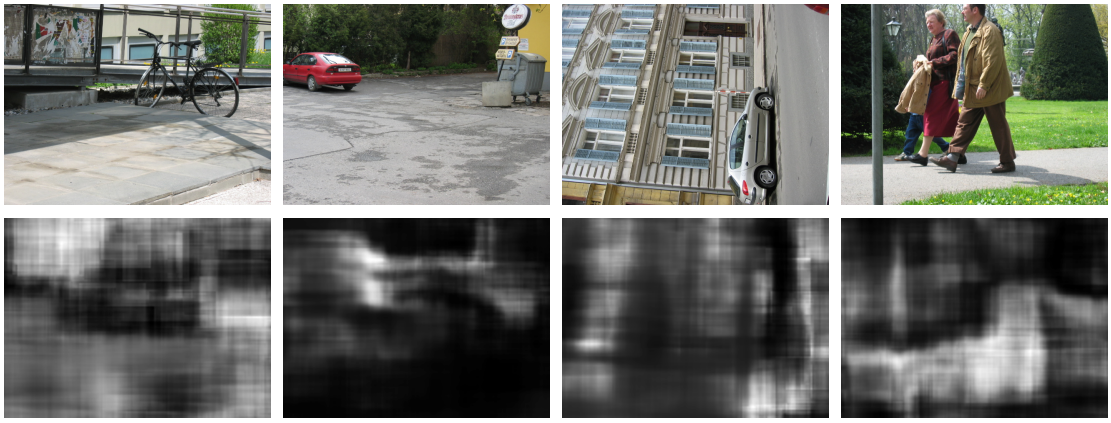


Figure 5.5. Some of the test images and the corresponding positive relevance weights, for which the relevance weighted classification was significantly outperformed by the conventional classifier.

Finally, for the images in figure 5.6, both classifiers fail at least eight out of ten times. For first two image, it appears that the relevance weights match the objects. Failure in these cases might be due to the limitation of the features used. With

addition features and descriptors,the classification accuracy might increase for these images. For the last two images the object size is small compared the image size, which leads to failure of the classifiers.

Figure 5.6. Some of the test images and the corresponding positive relevance weights, for which both the classifiers perform badly.

CHAPTER 6

CONCLUSION AND FUTURE WORK

This chapter reviews various contributions of the dissertation and suggests direction for continuation of the work in the dissertation.

6.1 Stereo Disparity Segmentation

In chapter 2, we proposed a novel iterative split-and-merge approach for the segmentation of the planar surfaces in the disparity space. The spatial continuity based splitting and the maximum allowable variance based merging were carried out iteratively to detect the number of planar surfaces and their corresponding parameters automatically. A new and efficient multi-stage merging algorithm based on the branch-and-bound search strategy was also proposed. The effectiveness of the proposed scheme and the branch-and-bound algorithm were demonstrated using experimental results for different data sets.

The quality of the results of the segmentation scheme depends on the quality of the stereo disparity. We use a basic disparity computation scheme to demonstrate the robustness of our method. Any of the disparity computation schemes available in the literature can be used to substitute the current computation scheme to the multi-stage branch-and-bound method. Additionally, the multi-stage branch-and-bound method can be modified to fit other clustering problems such as motion or image segmentation.

Regularization plays an important role in an inverse problem such as segmentation as well as disparity estimation. The smoothness and the spatial continuity are

commonly used in the disparity estimation for the regularization. Our method uses a bound on maximum allowable variance for the regularization. A better alternative to the sequential approach, in which the disparity computation is followed by the segmentation, is a simultaneous disparity estimation and segmentation approach. Such an approach would allow additional regularization criterions for both steps and would improve the segmentation result.

Compared to the quad-tree based spilt-and-merge, speedups are achieved by the proposed branch-and-bound algorithm in most of the cases. However, the algorithm can be improved with additional heuristics. Additionally, a combination of spatial segmentation with the proposed multi-stage merging algorithm might eliminate the need for the iterative process.

6.2 Two-View Multiple Structure and Motion Segmentation

We proposed a versatile new multiple structure-and-motion (MSaM) segmentation scheme and demonstrated its effectiveness through experiments in chapter 3. The branch-and-bound scheme can easily be scaled for parallel processing by solving each branch of the problem on a separate processor. Scheduling of these branches can be also an interesting direction of research. Although the method is proposed for a MSaM segmentation, it can be also applied to various other computer vision problems involving clustering such as segment based stereo [107], [108] and dense motion segmentation [13]. Since the outcome of the method heavily depends on the initial hypotheses chosen, various available guided sampling approaches have to be evaluated as to how well they explore and represent the solution space. The current approach can also be extended to an iterative approach. After each iteration of segmentation, fundamental matrices can be recalculated based on membership of the matches and these can be added as additional hypothesis to repeat the segmentation.

6.3 Computational Complexity of Branch-and-Bound

In chapter 4, we generalized the a branch-and-bound algorithm for model selection and analyzed its expected complexity. From the experimental results, the average complexity of the algorithm is much lower than the worst case complexity. Thus, branch-and-bound based model selection algorithms are practical for a hypothesis selection process, which has a moderate number of hypotheses and, when the size of optimal subset is small. With problem specific bounds and/or added heuristics, the computational complexity of the branch-and-bound algorithm can be improved further.

6.4 Visual Object Recognition

A positive and negative relevance based two-step classifier was proposed in chapter 5. The first step of the classifier establishes the relevance weights for image locations through a sliding window SVM classifier. The relevance weights are used to weight features during creation of histograms in the second step of the classifier. A significant improvement in the classification accuracies was observed with the proposed approach.

The accuracy of the classifier can be further improved by adding multiple region detectors and descriptors is the second step of the classifier. Also, other spatial techniques such as spatial pyramid matching [109] can also incorporated in the second step. As the first step of our classifier applies a sliding window based SVM classifier, it significantly slows down the overall classifier. To speed up the classification, a linear kernel combined with integral images/histograms technique can be used in the first step at the cost of some accuracy.

APPENDIX A

COST PROBABILITIES FOR UNIFORM IID

A sum of m uniform iids is distributed as,

$$f_m(S_m) = \frac{1}{(m-1)!}\sum_{j=0}^{m}(-1)^j\binom{m}{j}[(S_m - j)_+]^{m-1} \tag{A.1}$$

Similarly for a sum of n uniform iids,

$$f_n(S_n) = \frac{1}{(n-1)!}\sum_{k=0}^{n}(-1)^k\binom{n}{k}[(S_n - k)_+]^{n-1} \tag{A.2}$$

The corresponding cumulative distributions are,

$$F_m(S_m) = \frac{1}{m!}\sum_{j=0}^{m}(-1)^j\binom{m}{j}[(S_m - j)_+]^{m} \tag{A.3}$$

$$F_n(S_n) = \frac{1}{n!}\sum_{k=0}^{n}(-1)^k\binom{n}{k}[(S_n - k)_+]^{n} \tag{A.4}$$

For the above series involving S_m, in interval $j-1$ and j only j terms are non zero and for the series involving S_n, in interval $k-1$ and k only k terms are non zero.

Assuming S_m and S_n are independent, the joint distribution of S_m and S_n is product of the two. From the joint distribution of S_m and S_n,

$$\Pr(S_m < S_n) = \int_{S_n=0}^{n}\int_{S_m=0}^{S_n} f_m(S_m)f_n(S_n)dS_m dS_n \tag{A.5}$$

For $n < m$,

$$
\begin{aligned}
& \Pr(S_m < S_n) \\
= \; & \frac{1}{(n-1)!} \int_{S_n=0}^{n} \sum_{k=0}^{n} (-1)^k \binom{n}{k} [(S_n - k)_+]^{n-1} \\
& \left(\frac{1}{(m-1)!} \int_{S_m=0}^{S_n} \sum_{j=0}^{m} (-1)^j \binom{m}{j} [(S_m - j)_+]^{m-1} dS_m \right) dS_n \\
& \text{The bracketed expression is nothing but cumulative distribution of } S_m. \\
= \; & \frac{1}{(n-1)!} \int_{S_n=0}^{n} \sum_{k=0}^{n} (-1)^k \binom{n}{k} [(S_n - k)_+]^{n-1} \\
& \left(\frac{1}{m!} \sum_{j=0}^{m} (-1)^j \binom{m}{j} [(S_n - j)_+]^{m} \right) dS_n \\
& \text{Since } S_n \leq n, \text{ for } j > n \text{ the bracketed expression is 0,} \\
& \text{we change the upper limit of the summation over } j \text{ to } n. \\
= \; & \frac{1}{(n-1)!} \frac{1}{m!} \\
& \underbrace{\int_{S_n=0}^{n} \left(\sum_{k=0}^{n} (-1)^k \binom{n}{k} [(S_n - k)_+]^{n-1} \right) \left(\sum_{j=0}^{n} (-1)^j \binom{m}{j} [(S_n - j)_+]^{m} \right) dS_n}_{I}
\end{aligned}
\tag{A.6}
$$

We split integration I in to n segments of length 1. For $q = \{1, 2, \ldots, n\}$.

$$
\begin{aligned}
I_q &= \int_{S_n=q-1}^{q} \left(\sum_{k=0}^{q-1} (-1)^k \binom{n}{k} (S_n - k)^{n-1} \right) \left(\sum_{j=0}^{q-1} (-1)^j \binom{m}{j} (S_n - j)^{m} \right) dS_n \\
&= \sum_{k=0}^{q-1} \sum_{j=0}^{q-1} (-1)^{(k+j)} \binom{n}{k} \binom{m}{j} \underbrace{\int_{S_n=q-1}^{q} (S_n - k)^{n-1} (S_n - j)^{m} dS_n}_{I_q(j,k)}
\end{aligned}
\tag{A.7}
$$

Table A.1. Repeated differential table for $(t+j-k)^{n-1}$

p	Repeated differential
1	$(t+j-k)^{n-1}$
2	$(n-1)(t+j-k)^{n-2}$
3	$(n-1)\cdot(n-2)(t+j-k)^{n-3}$
p	$(n-1)\cdot(n-2)\ldots(n-(p-1))(t+j-k)^{(n-p)}$
$p+1$	$(n-1)\cdot(n-2)\ldots(n-(p-1))(n-p)(t+j-k)^{(n-(p+1))}$
$(n-1)$	$(n-1)\cdot(n-2)\ldots3\cdot2(t+j-k)$
n	$(n-1)\cdot(n-2)\ldots3\cdot2\cdot1$
$n+1$	0

Table A.2. Repeated integration table for t^m

p	Repeated integration
1	t^m
2	$\frac{t^{(m+1)}}{(m+1)}$
3	$\frac{t^{(m+2)}}{(m+1)(m+2)}$
p	$\frac{t^{(m+p-1)}}{(m+1)(m+2)...(m+p-1)}$
$p+1$	$\frac{t^{(m+p)}}{(m+1)(m+2)...(m+p-1)(m+p)}$
$(n-1)$	$\frac{t^{(m+n-2)}}{(m+1)(m+2)...(m+n-2)}$
n	$\frac{t^{(m+n-1)}}{(m+1)(m+2)...(m+n-2)(m+n-1)}$
$n+1$	$\frac{t^{(m+n)}}{(m+1)(m+2)...(m+n-1)(m+n)}$

Now we solve for integration $I_q(j,k)$.

$$
\begin{aligned}
I_q(j,k) &= \int_{S_n=q-1}^{q} (S_n-k)^{n-1}(S_n-j)^m dS_n \\
&\quad \text{Let } t=(S_n-j), \text{ then } dt=dS_n \text{ and } (S_n-k)=(t+j-k). \\
&\quad \text{When } S_n=q-1,\ t=q-1+j \text{ and when } S_n=q,\ t=q+j. \\
&= \int_{t=q-1+j}^{q+j} (t+j-k)^{n-1}t^m dt \qquad \text{(A.8)}
\end{aligned}
$$

We apply repeated integration by parts to (A.8) with help of the tables A.1 and A.2. After resubstituting $S_n - j$ for t we get,

$$\begin{aligned} & I_q(j,k) \\ = & \left[\sum_{p=1}^{n}(-1)^{(p-1)}\frac{[(n-1)\dots(n-(p-1))]\,(S_n-k)^{(n-p)}(S_n-j)^{(m+p)}}{(m+1)\dots(m+p-1)(m+p)}\right]_{(q-1)}^{q} \\ = & \left[\sum_{p=1}^{n}(-1)^{(p-1)}\frac{(n-1)!}{(n-p)!}(S_n-k)^{(n-p)}\frac{m!}{(m+p)!}(S_n-j)^{(m+p)}\right]_{(q-1)}^{q} \end{aligned} \tag{A.9}$$

Combining (A.6), (A.7), (A.8) and (A.9),

$$\begin{aligned} & \Pr(S_m < S_n) \\ = & \frac{1}{(n-1)!}\frac{1}{m!}\sum_{q=1}^{n}\sum_{k=0}^{q-1}\sum_{j=0}^{q-1}(-1)^{(k+j)}\binom{n}{k}\binom{m}{j} \\ & \left[\sum_{p=1}^{n}(-1)^{(p-1)}\frac{(n-1)!}{(n-p)!}(S_n-k)^{(n-p)}\frac{m!}{(m+p)!}(S_n-j)^{(m+p)}\right]_{(q-1)}^{q} \\ = & \sum_{q=1}^{n}\sum_{k=0}^{q-1}\sum_{j=0}^{q-1}(-1)^{(k+j)}\binom{n}{k}\binom{m}{j}\left[\sum_{p=1}^{n}(-1)^{(p-1)}\frac{(S_n-k)^{(n-p)}}{(n-p)!}\frac{(S_n-j)^{(m+p)}}{(m+p)!}\right]_{(q-1)}^{q} \end{aligned} \tag{A.10}$$

REFERENCES

[1] K. Konolige, "Small vision system: hardware and implementation," in *Eighth International Symposium on Robotics Research*, 1997.

[2] *Videre Design*, http://www.videredesign.com/.

[3] *Tyzx Inc.*, http://www.tyzx.com/.

[4] E. Izquierdo, "Disparity/segmentation analysis: matching with an adaptive window and depth-driven segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 4, pp. 589–607, 1999.

[5] Y. Altunbasak, A. Tekalp, and G. Bozdagi, "Simultaneous motion-disparity estimation and segmentation from stereo," in *Proceeding of IEEE International Conference on Image Processing*, vol. 3, 1994, pp. 73–77.

[6] D. Tzovaras, N. Grammalidis, and M. G. Strintzis, "Joint three-dimensional motion/disparity segmentation for object-based stereo image sequence coding," *Optical Engineering*, vol. 35, no. 1, pp. 137–144, 1996.

[7] S. Se and M. Brady, "Stereo vision-based obstacle detection for partially sighted people," in *Proceedings of Asian Conferance on Computer Vision*, 1997, pp. 152–159.

[8] K. Okada, S. Kagami, M. Inaba, and H. Inoue, "Plane segment finder: algorithm, implementation and applications," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 2001, pp. 2120–2125.

[9] E. Trucco, F. Isgro, and F. Bracchi, "Plane detection in disparity space," in *Proceedings of International Conference on Visual Information Engineering*, 2003, pp. 73–76.

[10] X. Wang and H. Wang, "Markov random field modeled range image segmentation," *Pattern Recognition Letters*, vol. 25, no. 3, pp. 367–375, 2004.

[11] F. Moscheni, S. Bhattacharjee, and M. Kunt, "Spatio-temporal segmentation based on region merging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 9, pp. 897–915, 1998.

[12] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[13] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing.*, vol. 3, no. 5, pp. 625–638, 1994.

[14] H. Nguyen, M. Worring, and A. Dev, "Detection of moving objects in video using a robust motion similarity measure," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 137–141, 2000.

[15] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

[16] R. Vidal and Y. Ma, "A unified algebraic approach to 2-D and 3-D motion segmentation," in *Proceedings of European Conference on Computer Vision*, 2004, pp. 1–15.

[17] A. Gruber and Y. Weiss, "Incorporating non-motion cues into 3D motion segmentation," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 84–97.

[18] M. Irani and P. Anandan, "A unified approach to moving object detection in 2D and 3D scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 577–589, 1998.

[19] H. Li, "Two-view motion segmentation from linear programming relaxation," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2007.

[20] R. Vidal and S. Sastry, "Optimal segmentation of dynamic scenes from two perspective views," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 281–286.

[21] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[22] E. Milios and E. G. M. Petrakis, "Shape retrieval based on dynamic programming," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 141–147, 2000.

[23] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 35, no. 1, pp. 13–32, 1999.

[24] T. B. Sebastian, P. N. Klein, and B. B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 550–571, 2004.

[25] D. Zhang and G. Lu, "Review of shape representation and description technique," *Pattern Recognition*, vol. 37, no. 1, pp. 1–19, 2004.

[26] L. Latecki, R. Lakamper, and T. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, 2000, pp. 424–429.

[27] D. Macrini, K. Siddiqi, and S. Dickinson, "From skeletons to bone graphs: medial abstraction for object recognition," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[28] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[29] A. Berg, T. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 26–33.

[30] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *Proceeding of IEEE International Conference on Computer Vision*, vol. 2, 2005, pp. 1458–1465.

[31] G. Wang, Y. Zhang, and L. Fei-Fei, "Using dependent regions for object categorization in a generative framework," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 1597–1604.

[32] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Proceedings of ACM International Conference on Image and Video Retrieval*, 2007, pp. 401–408.

[33] J. Corso, D. Burschka, and G. Hager, "Direct plane tracking in stereo image for mobile navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2003.

[34] N. Molton, S. Se, J. Brady, D. Lee, and P. Probert, "A stereo vision-based aid for the visually impaired," *Image and Vision computing*, vol. 16, no. 4, pp. 251–263, 1998.

[35] D. Burschka and G. Hager, "Scene classification from dense disparity maps in indoor environments," in *Proceedings of International Conference on Pattern Recognition*, vol. 3, 2002, pp. 708–712.

[36] B. Triggs, "Autocalibration from planar scenes," in *Proceedings of European Conference on Computer Vision*, 1998, pp. 89–105.

[37] Y. Kanazawa and H. Kawakami, "Detection of planar regions with uncalibrated stereo using distribution of feature points," in *Proceedings of British Machine Vision Conference*, vol. 1, 2004, pp. 247–256.

[38] M. Zuliani, C. Kenney, and B. Manjunath, "The multiransac algorithm and its application to detect planar homographies," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, 2005, pp. 153–156.

[39] M. Zucchelli, J. Santos Victor, and H. Christensen, "Multiple plane segmentation using optical flow," in *Proceedings of British Machine Vision Conferance*, 2002, pp. 313–322.

[40] Y. Ding, X. Ping, M. Hu, and D. Wang, "Range image segmentation based on randomized hough transform," *Pattern Recognition Letters*, vol. 26, no. 13, pp. 2033–2041, 2005.

[41] C. Veenman, M. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1273–1280, 2002.

[42] ——, "A cellular coevolutionary algorithm for image segmentation," *IEEE Transactions on Image Processing*, vol. 12, no. 3, pp. 304–316, 2003.

[43] M. Brusco and S. Stahl, *Branch-and-Bound Applications in Combinatorial Data Analysis*. Springer, 2005.

[44] K. Fukunaga, *Introduction to Statistical Pattern Recognition (2nd ed.)*. Academic Press Professional, Inc., 1990.

[45] P. Somol, P. Pudil, and J. Kittler, "Fast branch & bound algorithms for optimal feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 900–912, 2004.

[46] L.-K. Shark, A. A. Kurekin, and B. J. Matuszewski, "Development and evaluation of fast branch-and-bound algorithm for feature matching based on line segments," *Pattern Recognition*, vol. 40, no. 5, pp. 1432–1450, 2007.

[47] M. Roder, J. Cardinal, and R. Hamzaoui, "Branch and bound algorithms for rate-distortion optimized media streaming," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 170–178, 2006.

[48] J. Jonsson and K. Shin, "A parametrized branch-and-bound strategy for scheduling precedence-constrained tasks on a multiprocessor system," in *Proceedings of International Conference on Parallel Processing*, 1997, pp. 158–165.

[49] S. Fujita, M. Masukawa, and S. Tagashira, "A fast branch-and-bound algorithm with an improved lower bound for solving the multiprocessor scheduling problem," in *Proceedings of IEEE International conference on Parallel and Distributed Systems*, 2002, pp. 611–616.

[50] W. L. G. Koontz, P. M. Narendra, and K. Fukunaga, "A branch and bound clustering algorithm," *IEEE Transactions on Computers*, vol. 24, no. 9, pp. 908–915, 1975.

[51] N. Thakoor, V. Devarajan, and J. Gao, "Multi-stage branch-and-bound for maximum variance disparity clustering," in *Proceedings of International Conference on Pattern Recognition*, . 2008.

[52] N. Thakoor, J. Gao, and V. Devarajan, "Multistage branch-and-bound merging for planar surface segmentation in disparity space," *IEEE Transactions on Image Processing*, vol. 17, no. 11, pp. 2217–2226, 2008.

[53] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 1994.

[54] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[55] *Middlebury Stereo Vision Page*, http://www.middlebury.edu/stereo.

[56] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[57] K. Mühlmann, D. Maier, J. Hesser, and R. Männer, "Calculating dense disparity maps from color stereo images, an efficient implementation," *International Journal of Computer Vision*, vol. 47, no. 1, pp. 79–88, 2002.

[58] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*. Prentice-Hall, Inc., 2003.

[59] L. Vinet, "Segmentation et mise en correspondance de rgions de paires d'images stroscopiques," Ph.D. dissertation, Universit de Paris IX Dauphine, 1991.

[60] S. Chabrier, B. Emile, C. Rosenberger, and H. Laurent, "Unsupervised performance evaluation of image segmentation," *EURASIP Journal on Applied Signal Processing*, vol. 2006, no. 1, pp. 1–12, 2006.

[61] *JISCT Stereo Images*, http://vasc.ri.cmu.edu/idb/html/jisct/index.html.

[62] *Real-Time Stereo Vision based on the Uniqueness Constraint Experimental Results and Applications*, http://www.vision.deis.unibo.it/smatt/stereo.htm.

[63] A. D. R. McQuarrie and C.-L. Tsai, *Regression and Time Series Model Selection*. World Scientific, 1998.

[64] N. Thakoor and J. Gao, "Branch-and-bound hypothesis selection for two-view multiple structure and motion segmentation," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.

[65] T. T. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 2001.

[66] P. J. Huber, *Robust Statistics*. Wiley, 1981.

[67] P. Torr, *A structure and motion toolkit in Matlab*, http://cms.brookes.ac.uk/staff/PhilipTorr/Beta/torrsam.zip.

[68] K. Schindler, J. U, and H. Wang, "Perspective n-view multibody structure-and-motion through model selection," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 606–619.

[69] K. Schindler and D. Suter, "Two-view multibody structure-and-motion with outliers through model selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 6, pp. 983–995, 2006.

[70] R. Vidal, S. Soatto, Y. Ma, and S. Sastry, "Segmentation of dynamic scenes from the multibody fundamental matrix," in *Proceedings of Workshop on Visual Modeling of Dynamic Scenes in conjunction with European Conference on Computer Vision*, 2002.

[71] Y. Sugaya and K. Kanatani, "Multi-stage optimization for multi-body motion segmentation," *IEICE Transactions on Information and Systems*, vol. E87-D, no. 7, pp. 1935–1942, 2004.

[72] *The Hopkins 155 Dataset*, http://www.vision.jhu.edu/data/hopkins155/.

[73] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 2006.

[74] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[75] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 2008.

[76] S. Mitra and T. Acharya, *Data Mining: Multimedia, Soft Computing, and Bioinformatics*. Wiley-Interscience, 2003.

[77] S. Zhong and J. Ghosh, "A unified framework for model-based clustering," *Journal of Machine Learning Research*, vol. 4, no. 4, pp. 1001–1037, 2003.

[78] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[79] K. P. Burnham and D. Anderson, *Model Selection and Multi-Model Inference*. Springer, 2003.

[80] P. H. S. Torr, "Geometric motion segmentation and model selection," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 356, no. 1740, pp. 1321–1340, 1998.

[81] F. Glover and M. Laguna, *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc., 1993, ch. Tabu search, pp. 70–150.

[82] F. A. Chudak and D. B. Shmoys, "Improved approximation algorithms for the uncapacitated facility location problem," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 1–25, 2004.

[83] D. S. Hochbaum, Ed., *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1997.

[84] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[85] L. Devroye and C. Zamora-Cura, "On the complexity of branch-and bound search for random trees," *Random Structures and Algorithms*, vol. 14, no. 4, pp. 309–327, 1999.

[86] D. R. Smith, "On the computational complexity of branch and bound search strategies," Ph.D. dissertation, Duke University, 1979.

[87] ——, "Random trees and the analysis of branch and bound procedures," *Journal of the ACM*, vol. 31, no. 1, pp. 163–188, 1984.

[88] H. S. Stone and P. Sipala, "The average complexity of depth-first search with backtracking and cutoff," *IBM Journal of Research and Development*, vol. 30, no. 3, pp. 242–258, 1986.

[89] W. Zhang, "Branch-and-bound search algorithms and their computational complexity." University of southern California / Information sciences institute, Tech. Rep. ISI/RR-96-443, May 1996.

[90] W. Zhang and R. E. Korf, "Performance of linear-space search algorithms," *Artificial Intelligence*, vol. 79, no. 2, pp. 241–292, 1995.

[91] N. Thakoor, V. Devarajan, and J. Gao, "Computation complexity of branch-and-bound model selection," in *Proceedings of IEEE International Conference on Computer Vision*, . 2009.

[92] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*. Chapman & Hall/CRC, 2004.

[93] W. Feller, *An Introduction to Probability Theory and Its Applications*. Wiley, 1966, vol. 2.

[94] J. Willamowski, D. Arregui, G. Csurka, C. R. Dance, and L. Fan, "Categorizing nine visual classes using local appearance descriptors," in *Proceedings of Workshop on Learning for Adaptable Visual Systems in conjunction with International Conference on Pattern Recognition*, 2004.

[95] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: a comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[96] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[97] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[98] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[99] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[100] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[101] B. Fulkerson, A. Vedaldi, and S. Soatto, "Localizing objects with smart dictionaries," in *Proceedings of European Conference on Computer Vision*, 2008, pp. 179–192.

[102] J. C. Platt, *Advances in Large Margin Classifiers*. MIT Press, 2000, ch. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods, pp. 61–74.

[103] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer, "Generic object recognition with boosting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 416–431, 2006.

[104] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[105] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Proceeding of IEEE International Conference on Computer Vision*, 2009.

[106] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[107] M. Lin and C. Tomasi, "Surfaces with occlusions from layered stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1073–1078, 2004.

[108] L. Hong and G. Chen, "Segment-based stereo matching using graph cuts," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 74–81.

[109] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178.

BIOGRAPHICAL STATEMENT

Ninad Shashikant Thakoor received the Bachelor of Engineering degree in Electronics and Telecommunication Engineering from University of Mumbai in year 2001 and Master of Science degree in Electrical Engineering from University of Texas at Arlington in 2004. He received Ph.D. in Electrical Engineering at University of Texas at Arlington in December 2009. His research interests include shape classification, visual object recognition, branch-and-bound algorithms, stereo disparity segmentation and motion segmentation. He is member of IEEE and engineering honor society Tau Beta Pi.