Point, Line Segment, and Region-Based Stereo Matching for Mobile Robotics

Brian McKinnon Chi Tai Cheng John Anderson
Jacky Baltes
Dept. of Computer Science
University of Manitoba
Winnipeg, MB
Canada R3T 2N2
{ummckinno,umcheng,andersj,jacky}@cs.umanitoba.ca

Abstract

At the heart of every stereo vision algorithm is a solution to the matching problem - the problem of finding points in the right and left image that correspond to a single point in the real world. Applying assumptions regarding the epipolar rectification and color similarity between two frames is often not possible for real-world image capture systems, like those used rescue robots. More flexible and robust feature descriptors are necessary to operate under harsh real world conditions. This paper compares the accuracy of disparity images generated using local features including points, line segments, and regions, as well as a global framework implemented using loopy belief propagation. This paper will introduce two new algorithms for stereo matching using line segments and regions, as well as several support structures that optimize the algorithms performance and accuracy. Since few complete frameworks exist for line segment and region features, new algorithms that were developed during the research for this paper will be outlined and evaluated. The comparison includes quantitative evaluation using the Middlebury stereo image pairs and qualitative evaluation using images from a less structured environment. Since this evaluation is grounded in practical environments, processing time is a significant constraint which will be evaluated for each algorithm. This paper will show that line segment-based stereo vision with a gradient descriptor achieves at least a 10% *better accuracy than all other methods used in this evaluation while maintaining the low runtime associated with local feature based stereo vision.*

1 Introduction

The ability to perceive depth in real-time is an essential part of any mobile robotic platform. Information about the distance to objects in an environment is necessary for mapping, localization, and other high level tasks. Distance information is extracted using either active or passive sensors. Active sensors, such as laser radar (LADAR), computes

distances by emitting a signal into the environment and measuring the time it takes to reflect back to the sensor. Passive sensors, like cameras, simply collect the signals produced by the environment. When using a passive sensor, distance cannot be directly calculated since the time the signal takes to the reach the sensor is unknown. Multiple sensors can be used to extract distance information by locating the same signal source, or feature, in the environment and triangulating its position. Using two parallel cameras to observe an environment is referred to as stereo vision.

A stereo vision system must be able to extract features from an image and locate the corresponding features in a second image to perceive depth. The position difference or offset between a feature found in the first and second image is called the disparity. The disparity is measured horizontally and/or vertically for each matched pixel in the input images. The disparity values assigned to each pixel of the input images together produce a disparity image. Using the disparity image, a calibrated stereo system can triangulate the 3-D position of matched points in the environment.

Most stereo vision systems use point features that are located by passing filters over the input images. These filters produce a strong response at the corners and intersections of objects, or within textured regions (Shi and Tomasi, 1994). Point features are useful because the filters are very simple, and the extraction process has a very low computational cost. The problem is that points are difficult to match without significant constraints on the input images (Hollinghurst, 1997) due to a general problem with uniquely and accurately matching a point feature.

To reduce the uniqueness problem, more complex solutions that contain structural relationships can be used in the form of line segments, regions, and global frameworks. Line segments are extracted by clustering pixels located at the edge of the intensity change between two overlapping or neighbouring regions (Gonzalez and Woods, 2001). Regions are formed by clustering pixels together based on a similarity in colour and/or texture (Gonzalez and Woods, 2001) over a surface. Though regions and line segments contain more information than point features, they are computationally more expensive to extract from an image.

Global frameworks often model the stereo disparity extraction problem as a Markov random field (MRF) (Felzenszwalb and Huttenlocher, 2004). An MRF is an undirected graphical model connecting nodes of random variables. In the graphical model, nodes that are not connected have the Markov propery that they are conditionally independent random variables. Due to the connected nature of the problem the MRF contains loops in the graph that produce an optimization problem that is NP hard (Boykov et al., 2001). Two approximate solutions that produce reasonable results in practice are cut graphs (Boykov and Kolmogorov, 2004) and loopy belief propagation (Felzenszwalb and Huttenlocher, 2004). Global methods are known for producing dense high-quality depth maps even with a very simple pixel comparison method. The major downside is that global methods have a high cost in both memory usage and processing time.

The generation of accurate disparity images in real-time is the focus of this paper. Point features are the most widely used stereo feature, and have seen successful implementation in (Salari and Sethi, 1990; Se et al., 2005). Line segment-based stereo vision has been used primarily for uncalibrated stereo matching in (Bay et al., 2005; Zhang, 1995). Regions are primarily useful in environments when large areas of untextured surface are present and has been used in (Veksler, 2001; Bleyer and Gelautz, 2005). This

project involves adapting and developing algorithms for stereo matching that are capable of operating in real-time. Comparison of the three feature types and the global framework is done by comparing errors in the resulting disparity images for standard test images.

The problem of real-time stereo vision processing in this paper is grounded in the development of autonomous systems for urban search and rescue robots (USAR). Section 2 provides an overview of the principles of stereo vision and an examination of related work in feature point, line segment, and region extraction. The implementation details of the feature extraction and stereo matching problems are discussed in Sec. 3. Finally, the evaluation results will then be analyzed in detail by examining quantitative, qualitative, and runtime statistics (See. Sec. 4). The paper concludes with Sec. 5.

2 Related Work

Stereo vision is governed by a well defined relationship between the camera positions, the points in the environment, and the position on an image where the points are projected. The geometry governing the projection of stereo images will be examined first. Then existing work in the extraction and matching of points, line segments, and regions will be discussed including the benefits and limitations of each feature being examined. This will be followed by an examination of global algorithms using loopy belief propagation.

2.1 Stereo Vision

Passive vision sensors, like cameras, provide only 2-D information about objects in a 3-D world. However, when an object is viewed from multiple 2-D perspectives it is possible to triangulate the 3-D position. Offsetting the cameras produces a measurable disparity or offset of the pixels that is directly related to the distance of objects from the observer. To determine the 3D position of an object four problems must be solved (Lucas and Kanada, 1981):

1. Camera parameters must be determined,
2. Features must be identified in each image,
3. Corresponding features in each image must be matched correctly, and
4. The feature's distance must be calculated.

Using epipolar geometry (Hartley and Zisserman, 2004; Faugeras and Luong, 2001) it is possible to improve the performance of stereo vision systems by reducing the search space used in matching. Epipolar geometry (see fig. 1) defines a relationship between a point, x, on the image plane, P, of camera, C, to a line, e', on the image plane, P', of the second camera, C', known as an epipolar line. Epipolar lines represent the possible location of the image point, x', on P' for x on P that triangulates to the object point, M. An epipolar line on P' can be be visualized as the intersecting line between P' and a triangle with edges along the baseline joining C and C', the line joining C, x, and M, and the line joining M, x' and C'.

The reduction of the search space for a point in primary image from a 2-D region to a 1-D line in the secondary image is known as the epipolar constraint (Hartley and

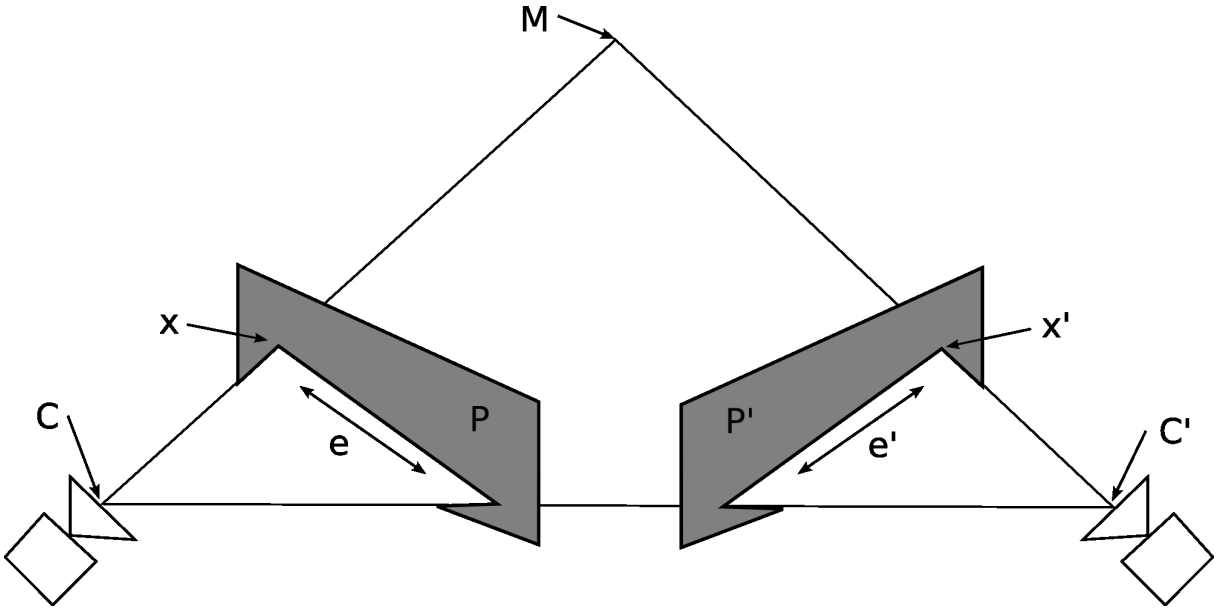

Figure 1: The epipolar line, e', represents the possible location of object point, M, on the image plane, P', of the secondary camera, C', given the image point, x, on the image plane, P, of the primary camera, C (Hartley and Zisserman, 2004).

Zisserman, 2004; Faugeras and Luong, 2001). Using the epipolar constaint images can be aligned so that the rows in the primary image are aligned with the rows in the secondary image. This allows matching to be performed without considering offsets in the vertical direction. The process of aligning the images is known as epipolar rectification.

Epipolar geometry is affected by external differences between the primary camera and secondary camera and the internal parameters of each camera. The external parameters include (Lucas and Kanada, 1981) the position and rotation difference from C' to C as shown in figure 1. The internal parameters include focal length, sensor width and height, and optional lens distortion parameters. If the internal and external parameters are known for the cameras it is possible to use the essential matrix (Faugeras and Luong, 2001) to compute the epipolar line. The essential matrix is defined as:

$$E = [t]_x R$$

where $[t]_x$ is a $3x3$ translation matrix and R is a $3x3$ rotation matrix. The $[t]_x$ matrix contains the position difference, or translation, between the cameras in the form:

$$[t]_x = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}$$

The essential matrix satisifies the epipolar contraint when:

$$\hat{x}'^T E \hat{x} = 0$$

where $\hat{x}$ and $\hat{x}'$ are the normalized image coordinates of x and x' computed using the calibrated internal parameters of the camera.

If the camera parameters are unknown then an approximation of the Essential matrix, known as the fundamental Matrix (Faugeras and Luong, 2001), can be used. Computing the fundamental matrix requires a minimum of eight matching points between the stereo images. These point matches require a high quality matching algorithm that produces accurate results without the use of the epipolar constraint.

Using the epipolar constraint it is possible to greatly increase both the speed and accuracy of any stereo matching system. The main issue is that calibration or unconstrained matching must be performed before the epipolar constraint can be used. In addition, the cameras must be very rigidly mounted since even small changes in the parameters of the cameras can produce large errors in the epipolar lines.

To calculate the distance to M, the only remaining variable is the conversion factor from image disparities to real-world distances. This can be determined by either measuring the baseline distance from C to C' or the distance from C to M. Methods for triangulating the position of a 3-D world point are described in (Trucco and Verri, 1998).

Point matching using epipolar geometry has been implemented in many prior works including (Urmson et al., 2002; Shibata and Kawasumi, 2004; Murray and Jennings, 1997; Sunyoto et al., 2004). Matching feature points between images is referred to as the correspondence problem (Salari and Sethi, 1990). Problems arise when a feature is distinct but not unique, occluded, or not fixed in the world. Non-unique points include texture features on a heavily textured surface that are distinct but repeated at other locations. The sidedness constraint (Bay et al., 2005) is often used in stereo matching to resolve the uniqueness problem. The constraint simply assumes that if a feature is located to the left of a second feature in the primary image, it will also be located to the left in the secondary image. Though the constraint is not always true, it is useful in many situations. Occlusion occurs when a feature is not visible in one of the images due to a foreground object overlapping a background object. Another problem with feature detection is that the feature must be fixed in the world. An example of an unfixed feature (Shi and Tomasi, 1994) can be found in a picture of a tree. The point where two branches cross produces a strong corner, but the point is not fixed so the position changes as the perspective changes.

2.2 Feature Points

A feature point (Shi and Tomasi, 1994) refers to a distinct fixed point in an image, and is primarily found in corners and textured areas (Shi and Tomasi, 1994). Each point in the image must be evaluated to determine its fitness as a feature point. Three highly successful methods of point feature extraction are Shi and Tomasi's eigenvalue features (Shi and Tomasi, 1994), Lowe's Scale Invariant Features (SIFT) (Lowe, 1999) and Bay, Tuytelaars and Van Gooland's Speeded Up Robust Features (SURF) (Bay et al., 2006).

The feature point extraction method used by Shi and Tomasi (Shi and Tomasi, 1994) is useful for extracting texture features and corners. The image is evaluated using a 2x2 matrix with the form:

$$Z = \begin{pmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{pmatrix}$$

where g_x and g_y are the horizontal and vertical gradient values. Gradient measures the first order change in the intensity of the pixel values over a defined kernel, and for images

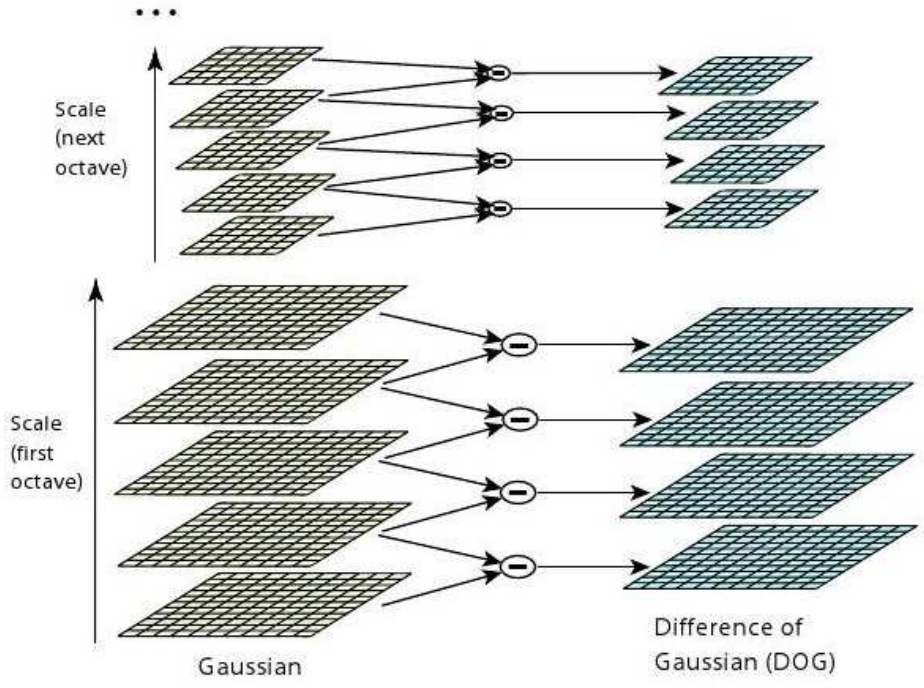

Figure 2: The Difference of Gaussian (DOG) used in SIFT is generated by subtracting sequences of Gaussian blurred images. Each new octave is created by shrinking the image to one quarter of the original size. Images from (Lowe, 1999).

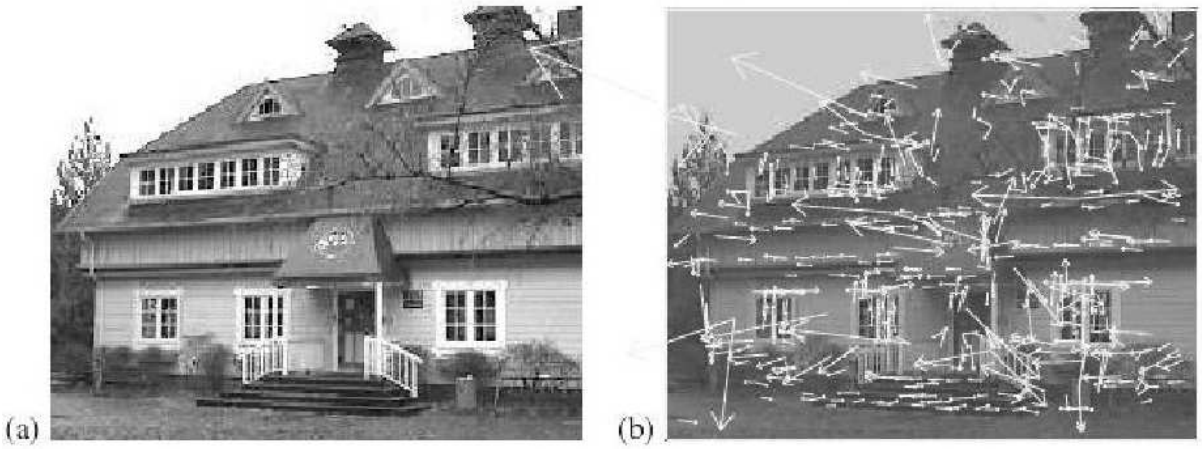

Figure 3: The image shows SIFT being used for feature detection. The arrows indicate the orientation and scale of the feature point. Images from (Lowe, 1999).

is computed horizontally and vertically. The two eigenvalues are then calculated for the matrix Z. Small eigenvalues indicate a flat colour, while one large and one small value indicate a horizontal or vertical line. Two large eigenvalues are generally found at corners or texture points.

SIFT extraction (Lowe, 1999) identifies strong features and the scale at which they

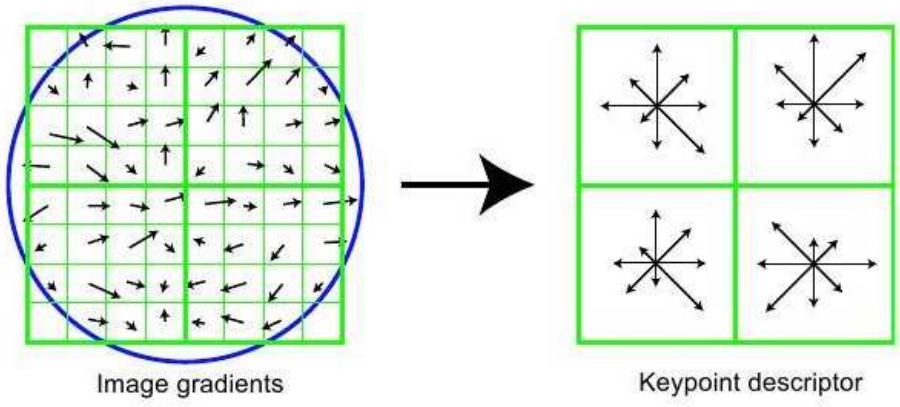

Figure 4: The feature descriptor histogram information is generated using gradients sampled around the feature point. Images from (Lowe, 1999).

produce the most distinct response. To locate points that remain visible at different scales, a series of Gaussian blurs and resizes are applied to the image. After each blur, the resulting image is subtracted from the previous image to produce a Difference of Gaussian (DOG) as shown in figure 2. From each pixel in the DOG, the centre pixel is selected as a base feature if it is the minimum or maximum difference for the 3x3x3 area around the point. The 3x3x3 are includes the current pixel and the neighbouring pixels in the previous, current, and next scale. The base set of features is initially filtered by removing points with low contrast using a threshold. The remaining points are filtered using the principle curvature, which is evaluated by computing the trace and determinant of a 2x2 Hessian matrix evaluated at the point. The Hessian matrix:

$$Z = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$$

where D_* is the derivative in the horizontal, vertical, and diagonal direction computed using the difference of neighbouring points. The DOG is repeated for several octaves of an image, where the next octave is reached by resizing the image to one quarter of the previous size. Orientation is assigned to the final points using the gradient response of sample points around the feature. The resulting orientation and scale is demonstrated in figure 3. A feature descriptor is generated by sampling a rotated area around the feature that aligns with the features orientation. Using 16 sample regions around the feature arranged in a 4x4 grid, an 8 bin gradient histogram is created per grid region as shown in figure 4. The gradient histogram contains the sum of the gradient responses in each grid region with longer vectors indicating strong responses in the direction shown. As a result of the sampling each feature contains a 128-dimensional descriptor that can be used to compare the features. When searching for a matching keypoint we expect the gradients histograms of all 16 grid regions to be very similar in magnitude between the two keypoints.

SURF (Bay et al., 2006) is a keypoint extraction method that makes use of fast approximations to produce a significant speed improvement over SIFT. The DOG filter in SIFT is replaced with a fast Hessian detector, $\mathcal{H}(x, \sigma)$, evaluated at the position x and

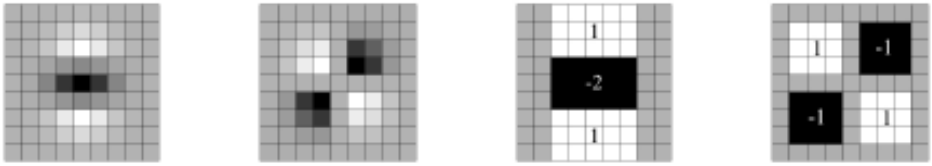

Figure 5: The Gaussian second order derivative can be approximated using box filters as shown above. Images from (Bay et al., 2006)

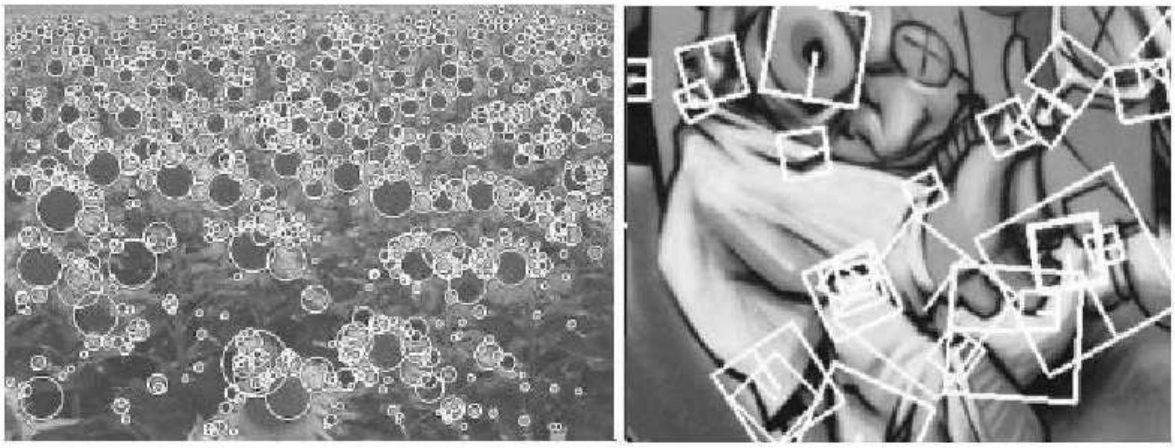

Figure 6: The images above shows a sample set of features generated by SURF. The left image shows a circle around the strongest keypoints. The image on the right shows the orientation and rotated sample region used to generate the descriptor. Images from (Bay et al., 2006)

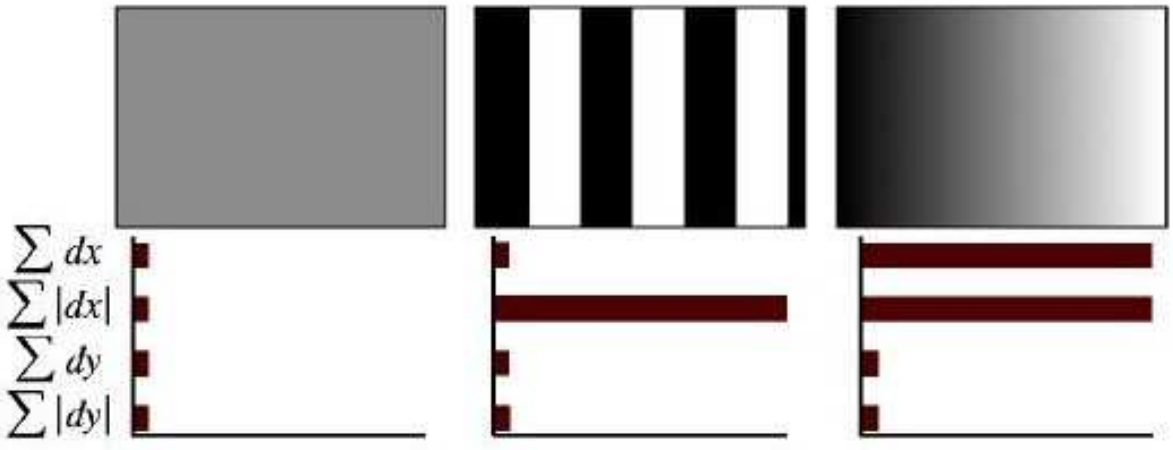

Figure 7: In SURF the feature descriptor information is generated by computing the sum of gradient and the sum of absolute gradient in the x and y direction. Images from (Bay et al., 2006)

scale σ. The Hessian matrix,

$$\mathcal{H}(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix}$$

where $L_{xx}(x,\sigma)$ is the Gaussian second order derivative in the horizontal direction, with $L_{yy}(x,\sigma)$, $L_{xy}(x,\sigma)$ being similar but in the vertical and diagonal direction. The fast Hessian detector takes advantage of the fact that the large scale convolutions can be extracted quickly from integral images using box filters. The box filters approximate the Gaussian using a rectangular convolution with constant weights for all elements as shown in figure 5. In the integral image the sum of all pixel values above and to the left of a pixel are accumulated into a single value. An integral image can be used to compute the response of a box filter in constant time regardless of the size of the region. Using an integral image reduces the computation cost by making convolution a constant time operation, and removing the need to resize the image during a change in octave. Points are selected if they contain the maximum or minimum response in a 3x3x3 area around the points and in the previous and next scale. The response is measured using,

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

where D is the box filter approximation of L. When a point is selected, the gradient is used to assign an orientation as demonstrated in figure 6. SURF use a 64-dimensional descriptor, that like SIFT are based on gradient responses around the feature. The descriptor uses a sixteen window grid around the point, but the histogram in SIFT is replaced with four values as shown in figure 7. The four values are generated using the sum of the gradient response and absolute gradient response in the x and y direction.

The key advantage of SIFT feature and SURF extraction is that they produce high quality feature descriptors at interest points of the image. In addition, the computation cost is predictable and can be controlled to some extent. For example, both SIFT and SURF are rotationally invariant, however, in stereo processing it can be assumed that a feature point will not rotate too much between the cameras. This allows for faster feature descriptor extraction by removing the orientation assignment and descriptor rotation phases of the feature extraction.

SIFT (Lowe, 1999) and SURF (Bay et al., 2005) use a high dimensional feature descriptor with responses between -1.0 and 1.0. Similarity between two features is generally measured using the Euclidean distance between the points in a high dimensional feature space (Beis and Lowe, 1997). The distance from each feature to all other existing features is computed, and a match is accepted if the best match has a distance less than the second best match distance scaled by a threshold ratio. The feature is compared against as many other features as possible to prevent a single point to point comparison from being accepted simply because there is not enough information to reject the match.

2.3 Line Segments

The extraction of line segments can be approached using either global or local methods (Jang and Hong, 2002). Both methods rely on identifying boundary pixels using one of several methods of edge detection. For more information refer to (Heath et al., 1997).

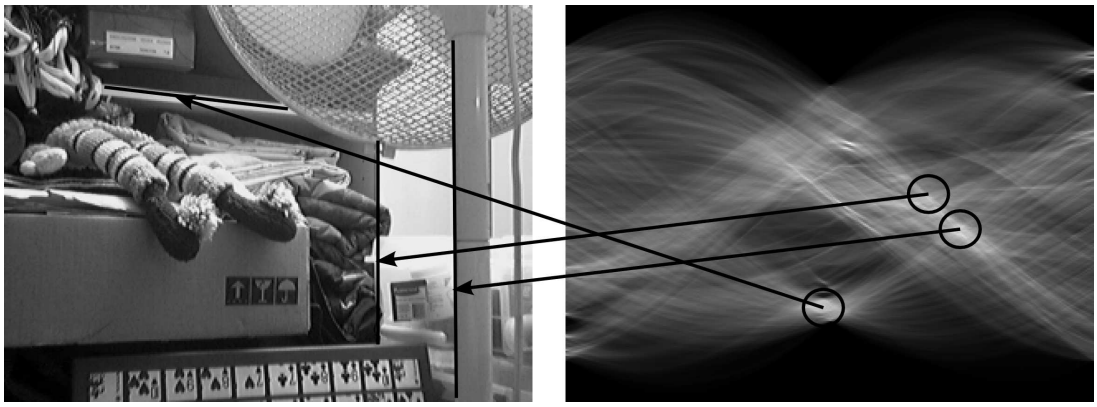

Figure 8: The left image is used to generate the Hough space shown on the right. The values in the Hough space are plotted using ϕ on the horizontal axis and ρ on the vertical axis. Points of high accumulation in the Hough space indicate strong lines in the image.

Most global line segment extraction algorithms are extensions of the generalized Hough transform (Kim et al., 2003). In the generalized Hough transform, each boundary pixel votes for candidate lines in the Hough space accumulator as shown in figure 8. Candidate lines are calculated using the line equation:

$$y * \sin(\phi) + x * \cos(\phi) = \rho$$

where x and y are the position of the pixel, ϕ is the orientation of a perpendicular line to the candidate line, and ρ is the length from the origin to the perpendicular intersection point. The accumulator is a two dimensional table that plots ϕ against ρ. For each ϕ value in the accumulator, ρ is calculated and the resulting point at (ϕ, ρ) in the accumulator is incremented. Once all boundary pixels are processed, a search is done in the accumulator to find peaks that exceed a threshold number of votes. Once a peak has been found the neighbouring bins in the accumulator are suppressed to prevent a similar line from being selected. This generalized Hough transform is used to extract lines from the image. Many other methods have been presented, for example (Kim et al., 2003; Mirmehdi et al., 1997), to deal with the extraction of line segments. The primary difficulty involved with global line segment extraction algorithms is that they have long running times, and can generate imaginary lines formed by textured surfaces.

The simplest form of local line segment extraction uses chain coding (Gonzalez and Woods, 2001). Chain code values represent the location of the next pixel in the chain using either a 4 or 8 connected neighbourhood. The boundary is followed starting from an initial edge point (generally the top-left most point in the chain) and followed until the chain returns to the start. Noise can be filtered from the chain code using a median filter over the connection direction. The final chain code is then examined for line segments by finding runs of equal values. Local line segment methods are more sensitive to noise in the image, therefore most of the recent work in this field focuses on joining small segments into larger segments (Kim et al., 2003; Jang and Hong, 2002).

Line segment matching is generally considered to be a more difficult problem than interest point matching. Although it seems simple to view line segments as simply a connected start and end point, the problem is that the position of the end points on line

segments tend to be very unstable (Bay et al., 2005). This makes end point matching more difficult than single interest point matching. The two primary features of a line segment used in matching are the colour information around the line segment, and the topology of line segments. Bay, Ferrari, and van Gool (Bay et al., 2005) use colour information from points three pixels to the left and right of a line segment to generate histograms. The histograms from the two candidate lines are normalized, and the distance between them is calculated using the Euclidean distance between histogram bins in the defined colour space. The colour information produces soft matches (Bay et al., 2005) reducing the number of potential matches. By applying the sidedness constraint (Bay et al., 2005), the incorrect matches are filtered out to produce the final set of matching lines. The use of colour information is limited to situations where the capture devices produced very similar colour output. In the more common situation, colour information varies between the images (Hollinghurst, 1997) due to automatic brightness and contrast correction differences between the capture devices.

If colour information is ignored, then the matching can be based on the topology of line segments in an image. Hollinghurst (Hollinghurst, 1997) has defined a set of rules for topological matching that includes initial matching and refinement operations. Using geometric constraints, initial matches are generated based on overlap, length ratio, orientation difference, and disparity limit. The initial matches are refined by applying constraints that either support or suppress a potential match. Matches are suppressed if they match multiple lines segments or if they violate the sidedness constraint. Support is generated by collinearity and connectivity. The forms of support that are important include parallelism, junctions at endpoints, and T-junctions. These constraints are applied to the initial matches until only supported matches remain.

Like point matches, line segments matching can benefit from the epipolar constraint. Zhang (Zhang, 1995) examined a method of calibrating stereo cameras by measuring the amount of overlapping area in line segment pairs for a given calibration. Calibration with line segments would be used in place of the eight point algorithm for point-based stereo matching. In his experiments, the extrinsic calibration for the two cameras were unknown, so the Essential matrix was estimated by using rotated points on an icosahedron. He found that given a set of matched line segments, the Essential matrix could be estimated as accurately as a fully calibrated trinocular system (Zhang, 1995) that was used as the ground truth in the experiment. One problem is that there is no discussion of the line segment matching problem, and it appears that prematched (possibly by a human operator) line segments were used. As a result, it is difficult to predict how line segment-based stereo vision calibration will work in the presence of noise found in real-world images.

2.4 Regions

The primary goal of region segmentation is to reduce the amount of information that must be considered by higher level processes. This is achieved by representing many pixels in an image as a single object. In real-world environments this is a very difficult problem, since it requires more than a simple clustering of similarly coloured pixels. This is due to the fact that objects often have inconsistent colouring, caused by textured surfaces and uneven lighting.

Figure 9: Results of scale-space watershed segmentation using Gaussian diffusion (Top) and CLMC (Bottom) to reduce over-segmentation. Images from (Vanhamel et al., 2003).

Though many region-based segmentation algorithms exist, patterned and heavily textured surfaces remain difficult to segment, since the boundary of a textured region is generally not well defined. Many approaches have been proposed including the use of fuzzy logic (Hanmandlu et al., 2004), wavelet transforms (Ardizzoni et al., 1999), discrete cosine transforms with Gabor filters (Kachouie et al., 2004), and Gabor filters with watershed transforms (Zhu and Basir, 2003). A key weakness of most of these methods is that they rely on supervision during the segmentation, because the number of textured regions must be precalculated for each image.

Researchers are attempting to address the problem of needing to know the number or regions by developing statistical analysis algorithms (Pok et al., 2004; Tay et al., 2002) that estimate the number of regions. These methods attempt to group sample areas from the image into texture bins. Then using the contents of the bins, analysis is done to determine the number of active textures. These methods often depend on a careful selection of parameters for the algorithm. For many application domains a simple region segmentation may produce reasonable enough results for practical usage.

Watershed segmentation (Gonzalez and Woods, 2001) is a popular form of region segmentation. Using the gradients of an image, local minima are selected as the seeding points for new regions. From the region seed points a hill climbing approach is used, where neighbours with a higher gradient are added to the region. The hill climbing is repeated until all the points in the image are assigned to a region. Watershed algorithms tend to over-segment an image due to noise in the captured processes. The effects of the over-segmentation can be reduced using a scale-space frameworks (Vanhamel et al., 2003), like Gaussian diffusion and CLMC, for merging neighbouring regions as shown in Figure 9.

A second method for segmenting images (McKinnon and Baltes, 2004) uses stack-based region growing with simple colour thresholding. The threshold values are small enough that regions only grow across a small portion of the image. These small regions are then merged into larger regions based on the amount of overlapping area. By growing small

Figure 10: The region segmentation (bottom) is produced from the raw image (top). The dot inside the regions represent the centroid point. Images from (McKinnon et al., 2005).

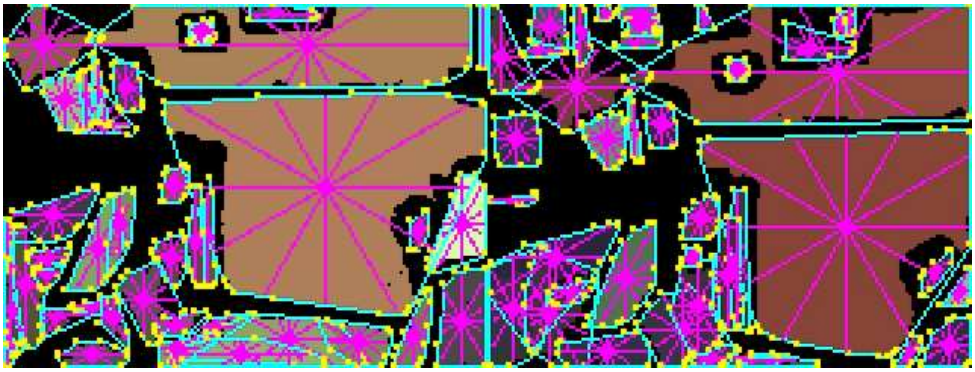

Figure 11: Convex hulls are used to represent the shape of regions. The hulls require few points so they reduce the memory and processing requirements of the region. The length of the lines radiating from the centre of mass are used as the hull signature. Images from (McKinnon et al., 2005).

portions of a region then merging the segmentation is able to extract regions that contain simple textures and uneven lighting. This method works well in many environments (see Fig. 10), since a large number of objects feature simple surface colour and texture. The key problem with this method is that the segmentation requires several iterations through the image and is therefore too slow to achieve real-time performance.

Segmented regions are initially represented by defining all the pixels that belong to the

region, which is inefficient for both memory usage and matching. If only the boundaries are stored, then the memory required to represent a region is greatly reduced. In an early version of the region segmentation considered for use in this paper (McKinnon et al., 2005), regions are reduced to a convex hull boundary using Graham's scan (Corman et al., 2001). The convex hull is then reduced to a signature consisting of a set of distance lines that radiate from the regions centre of mass to its hull boundary (see Fig. 11). The match quality for two regions is calculated using the sum of squared error in the signatures. The key benefit of the signature representation is its constant size, which allows for fast comparison. In addition, regions can be rotated quickly by simply shifting signature values to the left or right, allowing rotated regions to be compared just as quickly as upright regions. Region bounding with convex hulls has been tested and shown to be a good representation of simple objects, however, more complex regions containing non-convex boundaries are poorly represented (McKinnon et al., 2005).

2.5 Belief Propagation

Belief propagation (BP) is an iterative method for solving Markov random field (MRF) problems (Felzenszwalb and Huttenlocher, 2004). Solutions to the stereo matching problem use a special case of BP known as loopy BP (LBP) since the message passing described below includes cycles or loops in the nodes of the MRF. As an initial note, calculations are performed using negative log probabilities, since max-product equations are replaced with min-sum, which improves numerical stability. Using a set of finite labels, $\mathcal{L}$, and pixels, $\mathcal{P}$, a label $f_p \in \mathcal{L}$ is assigned to each pixel $p \in \mathcal{P}$. The goal is to compute the quality of a labeling at each pixel which maximizes the energy function,

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} W(f_p, f_q) \tag{1}$$

where $\mathcal{N}$ is the set of edges connecting the node to its neighbouring nodes. As equation 1 shows, the goal is to compute a set of labels that maximize the data match, $D_p(f_p)$, as well as maintains smoothness, $W(f_p, f_q)$, across neighbouring nodes. The optimal labeling for each node is computed using an iterative message passing scheme that propagates information through the MRF. The messages, $m^t_{p \to q}(f_q)$, from p to q are computed using the function,

$$h(f_p) = D_p(f_p) + \sum_{s \in \mathcal{N}(p)/q} m^{t-1}_{s \to p}(f_p) \tag{2}$$

$$m^t_{p \to q}(f_q) = \min_{f_p} \left(V(f_p - f_q) + h(f_p) \right) \tag{3}$$

where $\mathcal{N}(p)/q$ is the neighbours of p excluding q, and $V(f_p - f_q)$ is the discontinuity cost between labels in the node. The discontinuity cost can be computed in a variety ways including the Potts, linear or quadratic model using the distance between neighbouring labels. The message passing is run for a maximum number iterations or until convergence, however, convergence is not guaranteed to occur in LBP (Yedidia et al., 2003).

LBP produces good solutions in practice, but at a high cost in both memory and computation. In (Felzenszwalb and Huttenlocher, 2004), LBP for stereo is implemented using disparity as the set of labels. The data cost for a label is computed using the absolute difference of intensity for a pixel in the first image from a pixel offset by the

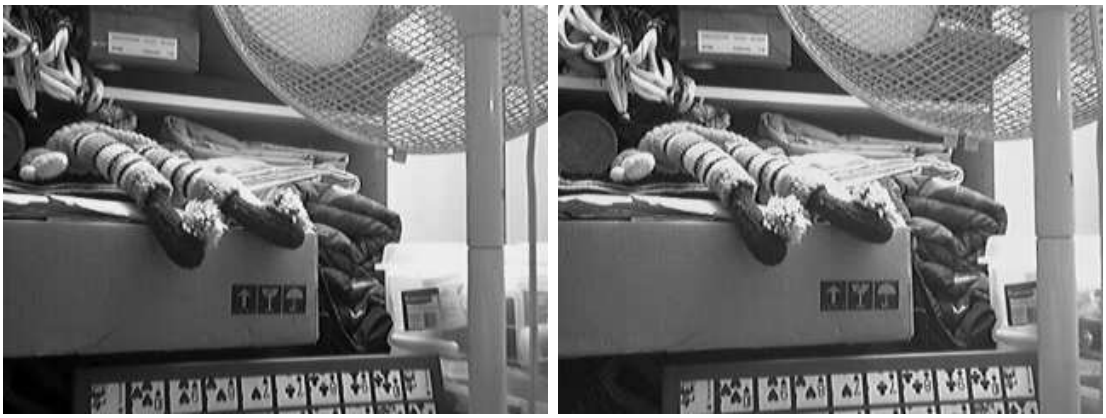

Figure 12: The left and right input images used in the description of the algorithm.

label's disparity in the second image. When the discontinuity cost is linear the LBP algorithm can distribute the cost in linear time, since information is exchanged only with neighbouring labels processed left to right then right to left. The computation time of the LBP implementation in (Felzenszwalb and Huttenlocher, 2004) is $O(nkT)$ where n is the number of pixels, k is the number of labels, and T is the number of iterations. To reduce T, an image pyramid is used that propagates belief across a coarse to fine resolution version of the MRF. It should be noted that memory requirements are significant, with $5nk$ floating point values required to process the LBP. This means that a 320x240 image with 64 labels will require just under 100MB of memory to store the finest resolution of the graph.

2.6 Summary

As the literature review above demonstrates, there is a significant amount of development being done in the area of stereo depth extraction. Though global algorithms like BP and local features like SIFT and SURF have been used extensively there is still a large gap in the number of complete solutions using line segments and regions. The implementation details in the next chapter outline the research work that has been done in this paper to explore new methods of local feature-based stereo depth extraction.

3 Implementation

My approach to feature-based stereo matching involves the development of two new feature extraction and stereo matching components. The line segment and region-based extraction and matching developed and implemented during this paper project will be explained in detail. The preprocessing stage of the line segment and region-based algorithms are the same, so they will be discussed before the feature specific details.

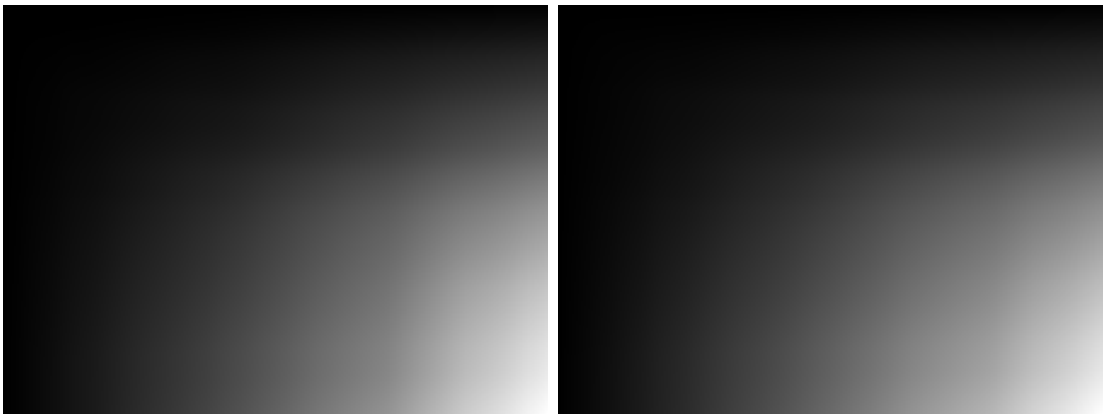

Figure 13: The integral images generated for the left and right input images.

3.1 Preprocessing for Line Segment and Region Extraction

In this section, the preprocessing algorithms for the line segment and region extraction will be outlined. The steps of the preprocessing algorithm are:

1. Integral Image Generation
2. Haar Feature Extraction
3. Gradient Thinning

For simplicity, speed, and grounding, the algorithms described in this implementation will be examined using the 8-bit gray-scale images of dimensions 320x240 shown in figure. 12.

The integral image is an efficient data structure for extracting large amounts of data from images in constant time. In an integral image, the value stored in each pixel is the sum of all intensities in the area above and to the left of that pixel, as shown in figure 13. This means that the sum of any rectangular subregion inside the integral image can be extracted with only four table lookups and three addition/subtraction operations. For example, integral images can be used to apply a blur to an image with a simple box kernel. This is achieved by taking the summed area in a rectangular region around each point and normalizing by the area of the rectangle. The benefit of this method over conventional blurs is that runtime is independent of the kernel size and, unlike separable kernels, requires no additional temporary storage.

Another application of integral images is to extract arbitrarily sized Haar features from an image. Haar features a contructed using multiple box filters as shown in figure 14, and unlike a Gaussian filter provide a constant weight for each sample in the filter. There are several Haar features that can be extracted from an image, however, this algorithm focuses on vertical and horizontal gradient features. During the Haar feature extraction stage of the algorithm, the two gradient features are extracted at a kernel size of 4x4 (see figure 15), 8x8, and 12x12 pixels. The values are cached into a lookup table to improve performance during the matching stage. These features then provide the input for both gradient thinning and matching.

The gradient thinning algorithm is based on the Canny method (Canny, 1983), with the addition of the Haar feature gradients and an efficient data structure. The goal of this

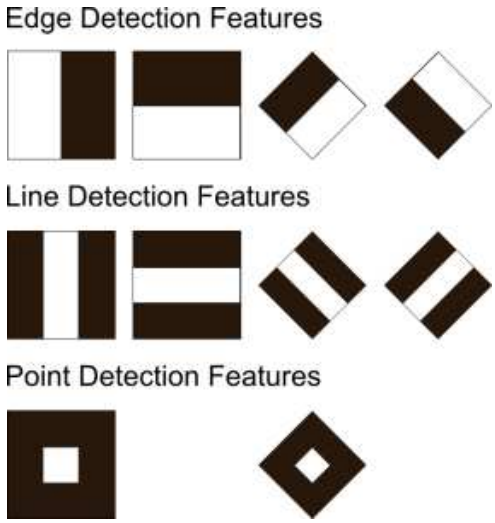

Figure 14: A partial set of Haar features used in the extraction of data from integral images. The sum in the black area is subtracted from the sum in the white area. This set includes both up-right and diagonal features.

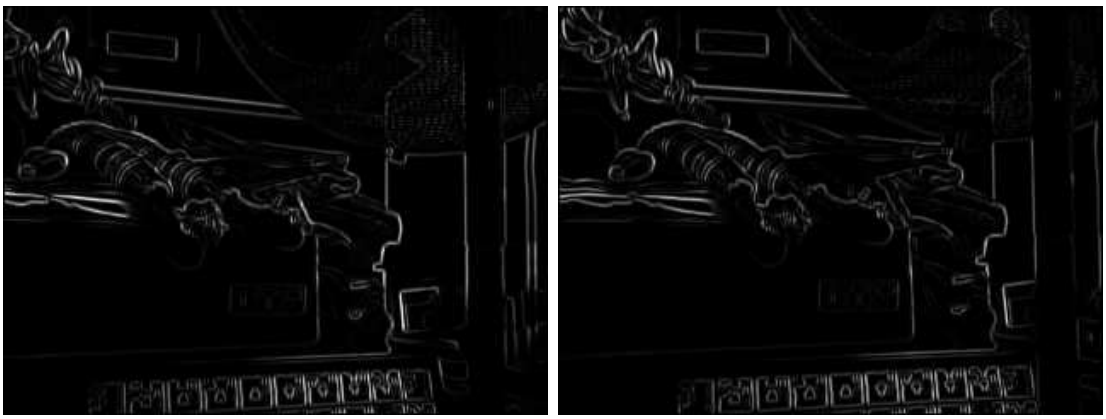

Figure 15: The gradient magnitude of the left and right input images.

method is to extract the peak points by keeping only the strongest gradient magnitudes along the gradient orientation. From the cached Haar features, one kernel size is selected for gradient thinning with the most common choice being 4x4. The points are initially thresholded using the magnitude of the gradient and are then used to activate an edge cell (EC).

The EC is an efficient data structure for storing gradient orientations as shown in figure 16, since it allows binary comparisons of the gradient orientation. Each EC is a dual layer ring discretized into 8 cells, allowing a resolution of 45 degrees per ring. The inner layer is offset from the outer layer by a distance of 22.5 degrees, and the combined layers produce a 22.5 degree resolution for the EC. The gradient orientation of each pixel activates one cell in each layer of the pixel's EC. Using this activation, the EC can be quickly tested for a horizontal, vertical, or diagonal orientation with a binary logic operation. ECs can be compared against each other using either an equal

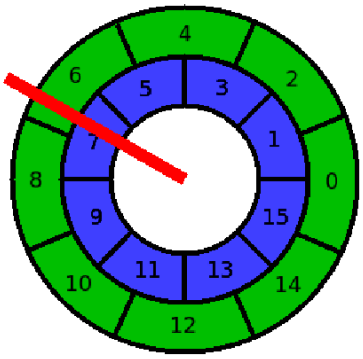

Figure 16: The Edge Cell (EC) binary data structure for storing orientation information. The key benefit of the EC structure is that it allows fast binary comparisons of the gradients.

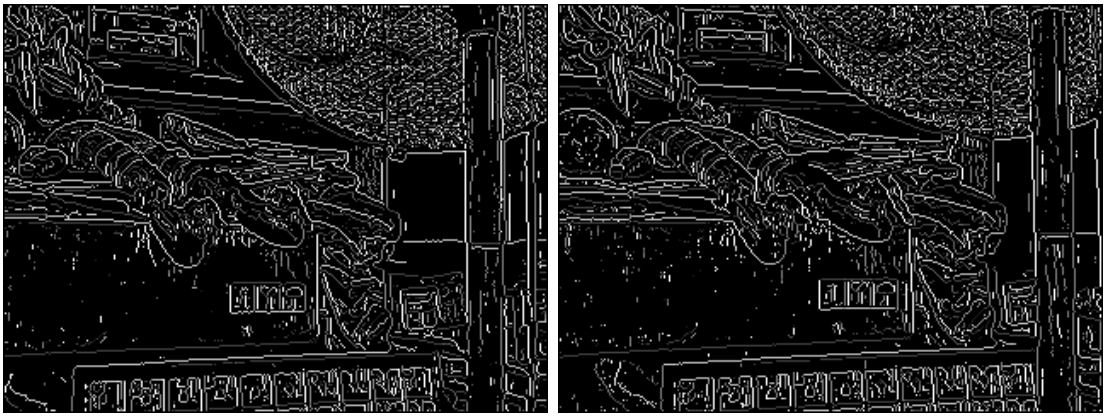

Figure 17: The edge cell activation of the points remaining after gradient thinning.

operator, which returns true if both cell layers overlap, or a similar operator, which returns true if at least one cell layer overlaps. The thinning direction is determined by the EC activation, and only the strongest gradient points (see figure 17) along the thinning orientation are processed further. Using the EC, a thinning direction can be computed using two operations. For example, the EC shown in figure 16 can be tested to see if it is diagonal in one binary operation, and since it is true it would then be tested to check if it is on the forward slash diagonal, which returns false. Without the EC the fastest way to check the thinning direction would be to use a binary search through the floating point gradient orienation value. In most environments the binary comparisons should execute faster then the floating point comparisons.

3.2 Line Segment Extraction and Matching

In this section, the line segment extraction and matching algorithm will be outlined. The steps specific to this algorithm are:

1. Binary Segmentation

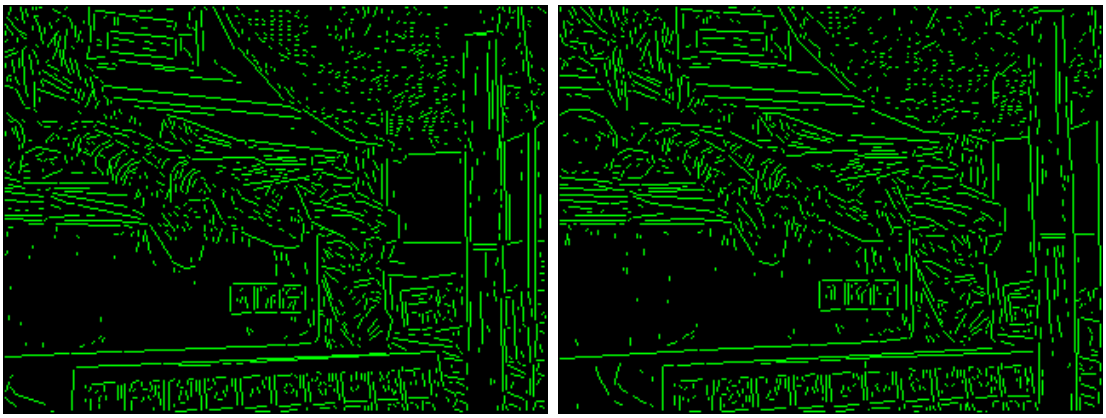

Figure 18: The final line segments extracted by the described algorithm.

2. Overlap Clean-up

3. Dynamic Program Matching

The binary segmentation is the core of the line segment extraction and makes use of the EC data structure described above. For each active pixel, the EC is separated into two Split ECs (SEC) with the first containing the inner ring activation, and second containing the outer ring activation. An 8-neighbour binary segmentation (Gonzalez and Woods, 2001) then tests each of the two SECs separately against its neighbours. The binary test for the neighbour returns true if that neighbour EC is similar (at least one cell overlaps) to the current SEC. For example, the EC shown in figure 16 is split into one SEC with activation only in bit number 6, and a second SEC with activation only in bit 7. During the segmenation the SEC with bit number 6 activated will join with neighbouring ECs that have bit 5 and 6, or 6 and 7 activate. The final result of the segmentation is that each pixel is a member of two thinned binary regions, or line segments as they will be referred to from now on. The overlapped line segments are useful, but we generally prefer that each pixel only belong to a single line segment.

There are two cases of overlap that need to be cleaned up to produce the final result. The first clean-up step is the removal of any line segments contained mostly within another longer line segment. This is achieved by simply having each pixel cast a vote for the longer of the two line segments for which it is a member. Any line segment with a number of votes below a threshold is discarded. The second clean-up step involves finding an optimal cut point for any partially overlapping line segments. This is achieved by finding the point in the overlapping area that maximizes the area of a triangle formed between the start of the first line segment, the cut point, and the end point of the second line segment. The overlapping line segments are then trimmed back to the cut point to produce the final line segment. Any line segments smaller than a defined threshold are discarded at the end of this stage. The final set of line segments (see figure 18) are then used in the matching process.

The matching of line segments is a difficult task since the end points are generally poorly localized and the segments can be broken into multiple pieces due to noise in the image. In addition, some line segments may not be completely visible in both images

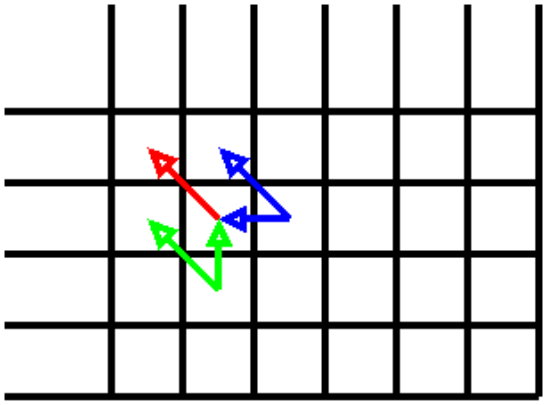

Figure 19: Entries along the diagonal may only move along the diagonal, while those outside the diagonal may move towards or parallel to the diagonal.

due to occlusion. To address these problems, a dynamic programming (DP) solution is applied to the line segment matching problem. This implementation will be defined as DP Line Segment Descriptor Matching (DPLSDM). The DP table consists of points from the source line segment, ls_1, making up the row header, and the points from the matching line segment, ls_2, making up the column header. The goal is to find the overlapping area that maximizes the match value between the points of the two line segments. To compare the points of two line segments, we return to the Haar features extracted earlier. The match value for two point feature vectors, v_1 and v_2, is calculated as the sum of the feature vector match function.

Algorithm 1: match_feature_vector

Input: Feature vector v_1 and v_2
Output: The matching value for the features, R
$M = 0, T = 0$;
foreach *Dimension* i *in* v_1 *and* v_2 **do**
  $P = v_{1i} + v_{2i}$;
  if $P >= 0$ **then**
    $M = M + \min(v_{1i}, v_{2i})$;
    $T = T + \max(v_{1i}, v_{2i})$;
  end
  if $P < 0$ **then**
    $M = M + \min(-v_{1i}, -v_{2i})$;
    $T = T + \max(-v_{1i}, -v_{2i})$;
  end
end
$R = M/T$

Each insertion into the table involves selecting a previous match value from the table, adding the current match value, R, and incrementing a counter for the number of points contributing to the final result.

The insertion of the match values into the dynamic programming table requires certain assumptions to prevent the table from becoming degenerative. The assumptions are defined algorithmically using the variables x for the current column, y for the current

row, Dp for the dynamic programming table, St for the match sum table, Ct for the point count table, and R for the current match value. The Dp value is generated from St/Ct and for simplicity the assignment $Dp[x][y] \leftarrow Dp[x-1][y-1]$ is actually $St[x][y] = St[x-1][y-1]+R$ and $Ct[x][y] = Ct[x-1][y-1]+1$. With that in mind, the assumptions used to generate the Dp table are:

1. To prevent oscillation, if a match diverges from the centre line it cannot converge back to the centre line. The entries in the table sample previous entries as shown in figure 19, and is enforced using the algorithm 2.

Algorithm 2: compute_match

Input: The empty line matching dynamic programming table Dp for ls_1 and ls_2
Output: The completed line matching dynamic programming table Dp for ls_1 and ls_2

```
foreach Element, x, in the ls1 do
    foreach Element, y, in the ls2 do
        if x = y then
            Dp[x][y] ← Dp[x − 1][y − 1];
        end
        if x > y then
            Dp[x][y] ← compute_best(Dp[x − 1][y − 1], Dp[x − 1][y]);
        end
        if x < y then
            Dp[x][y] ← compute_best(Dp[x − 1][y − 1], Dp[x][y − 1]);
        end
    end
end
```

2. No point in the first line can match more than two points in the second line. This prevents the table from repeatedly using one good feature to compute all matches, and is enforced by defining the behaviour of the *compute_best* function using algorithm 3. This assumption enforces a general belief that two matching line segments will have a comparable length in the range of 50% to 200% in the number of points in each line segment.

Algorithm 3: compute_best

Input: Element index $x1$, $y1$ in ls_1, and $x2$, $y2$ in ls_2
Output: The best match results from either $x1$, $y1$ or $x2$, $y2$

```
if Ct[x1][y1] > Ct[x2][y2] + 1 then
    return Dp[x2][y2];
end
if Ct[x1][y1] + 1 < Ct[x2][y2] then
    return Dp[x1][y1];
end
return max(Dp[x1][y1], Dp[x2][y2]);
```

The best match value is checked in the last column for all rows with an index greater than or equal to the length of ls_2, or for any entries in Ct with a count equal to the

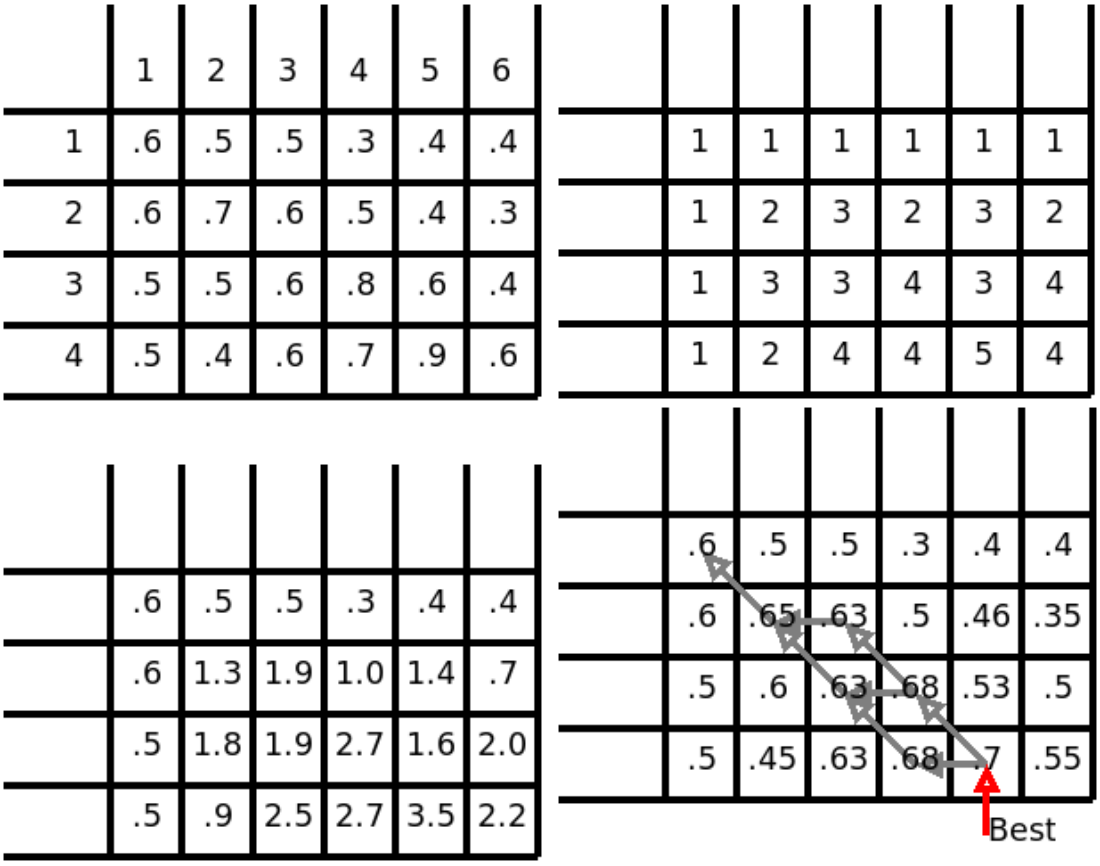

Figure 20: An example match table is shown in the top-left. The count table (Ct) (top-right) and sum table (St) (bottom-left) are generated iteratively using algorithms 1, 2, and 3. The resulting dynamic programming (bottom-right) table is generated by combining the St and Ct table. From the best matching entry in the outer edge of the table, the best path is back traced through the dynamic programming table by selecting the best previous entry using the same direction rules that generated the table.

size of ls_1. Once a best match value is found, a back trace is done through the table to find the point match that produced the best match. The position difference between the matched points in ls_1 and ls_2 is used to determine the disparity of the matched pixels. Linear regression is used to generate a disparity function that maps a point in the line to a disparity value for each point in ls_1. The best match value is then recalculated using the disparity generated by the linear regression line, since the best match value should be based on the final disparity match. The matching process is repeated for all ls_2 segments in the potential match set. A matching example is shown in figure 20.

To reduce the number of line segments that are compared in the DPLSDM, a few simple optimizations are performed. The primary filtering is achieved by only comparing line segments that have similar edge cell activations. The secondary method is to apply a threshold to the maximum search space, and this is done in two ways. The first threshold involves placing a bounding rectangle extended by the size of the search space around ls_1, with a second rectangle placed around ls_2. If the two bounding rectangles overlap, then the comparison continues. The second threshold is done on a point-by-point basis, with

points outside the search space receiving a match value of zero. These simple optimizations greatly increase the speed of the algorithm, making real-time performance achievable.

A final filtering process is applied in the DPLSDM to account for line segments that have no exact match in the second image. Noise during the line segment extraction can cause a segment to be split into multiple pieces making the matching process more error prone. To address this problem, the best disparity functions generated by the linear regression at the end of the matching process is stored and then applied to each line segment once more. A matching score for the line segment is generated by applying the disparity function to each point in the line segment and computing the error between the point in the first image and the displaced point in the second image using algorithm 1. The disparity function that produces the lowest match score is used to generate the final disparity image.

3.3 Region Extraction and Matching

The region extraction method is adapted from a path planning algorithm using Flexible Binary Space Partitioning (FBSP) (Baltes and Anderson, 2003). The adapted FBSP separates an image into regions of occupied and free space by recursively cutting the image into subregions at a cutting line. The gradient thinned edge activation is used to identify occupied and free space in the image. Any active EC in the thinned image will be flagged as occupied, with all other points assigned as free space. The cutting line is selected by computing the entropy and information gain, at each row and column in the current region.

Algorithm 4: compute_entropy

Input: The percentage of free, P_f, and occupied, P_o, space
Output: The entropy, E, of the subregion
$E = -(P_f * \log_2(P_f)) - (P_o * \log_2(P_o))$;

Algorithm 5: compute_gain

Input: The entropy, E, of regions RB, $R0$, and $R1$
Output: The information gain, G, of a region cut
$G = E_{RB} - (area(R0) * E_{R0} + area(R1) * E_{R1})/area(RB)$;

Entropy and information gain is computed using the percentage of free, P_f, and occupied, P_o, space resulting from the candidate cutting line. The entropy E of a subregion is calculated using algorithm 4. The information gain, G, for a base region, RB, and two subregions, $R0$ and $R1$, created by splitting RB at the cut line is calculated using algorithm 5. The cut that produces the highest information gain is used if the gain is greater than zero. Using an integral image generated from the gradient thinned edge activation, P_f and P_o for each region can be computed in constant time for all possible cut lines. A sample segmentation using this method is shown in Figure 21.

Stereo matching is achieved using the same feature descriptor and matching function as the DPLSDM algorithm 1. The difference in FBS Descriptor Matching (FBSDM) is where the feature comparison is done. Initial investigations indicated that matching cut points in the left and right FBSP results produced unreliable results, since the cutting order in the FBSP was inconsistent between similar images. This led to a matching solution that required an exhaustive evaluation of the search space. It was thought that

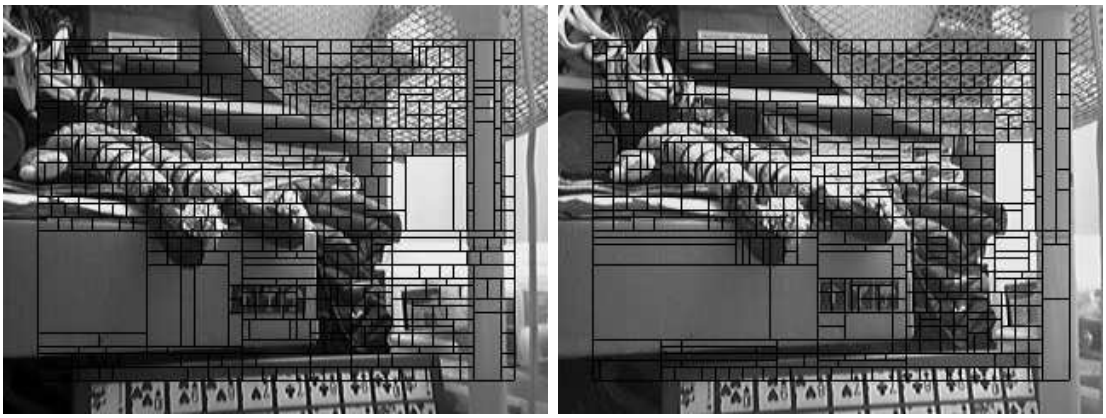

Figure 21: The regions selected from the test image using flexible binary space partitioning.

the regions could be matched by creating a feature descriptor contained within the entire boundary of the region. This matching method, however, produced poor results due to occlusions in the image. To achieve more robust results, feature points are selected at the four corner points and the centre of a region. From the five points, the four strongest matches at each disparity in the search space are used to produce the final match result. The disparity that produces the strongest matching result is assigned to all pixels in the region.

4 Evaluation

This section describes the results of our evaluation of the described algorithms on the Middlebury data set as well as from real-life video footage from our robots.

4.1 Middlebury Data Set

The Middlebury data set (Scharstein and Szeliski, 2002) is considered a standard in the evaluation of dense stereo vision algorithms. The data set was expanded in (Scharstein and Szeliski, 2003; Scharstein and Pal, 2007) to include more stereo pairs. The key advantage of the stereo pairs provided in the Middlebury set is that they include both input images and dense disparity maps. There are several tests defined in (Scharstein and Szeliski, 2002) for comparison of dense two frame stereo methods. These tests include the total number of matching pixels in the entire image, in areas of occlusion, and in textureless regions. The area of occlusion test targets regions that are only visible from a single image. The textureless regions provide information about the ability of an algorithms to in-fill areas with no matchable information. These tests are not well suited for semi-dense stereo matching, since they focus on areas with little or no matchable information. Unlike dense algorithms, semi-dense methods focus on areas of maximum information density in the image. Therefore, a different performace metric is used in this evaluation.

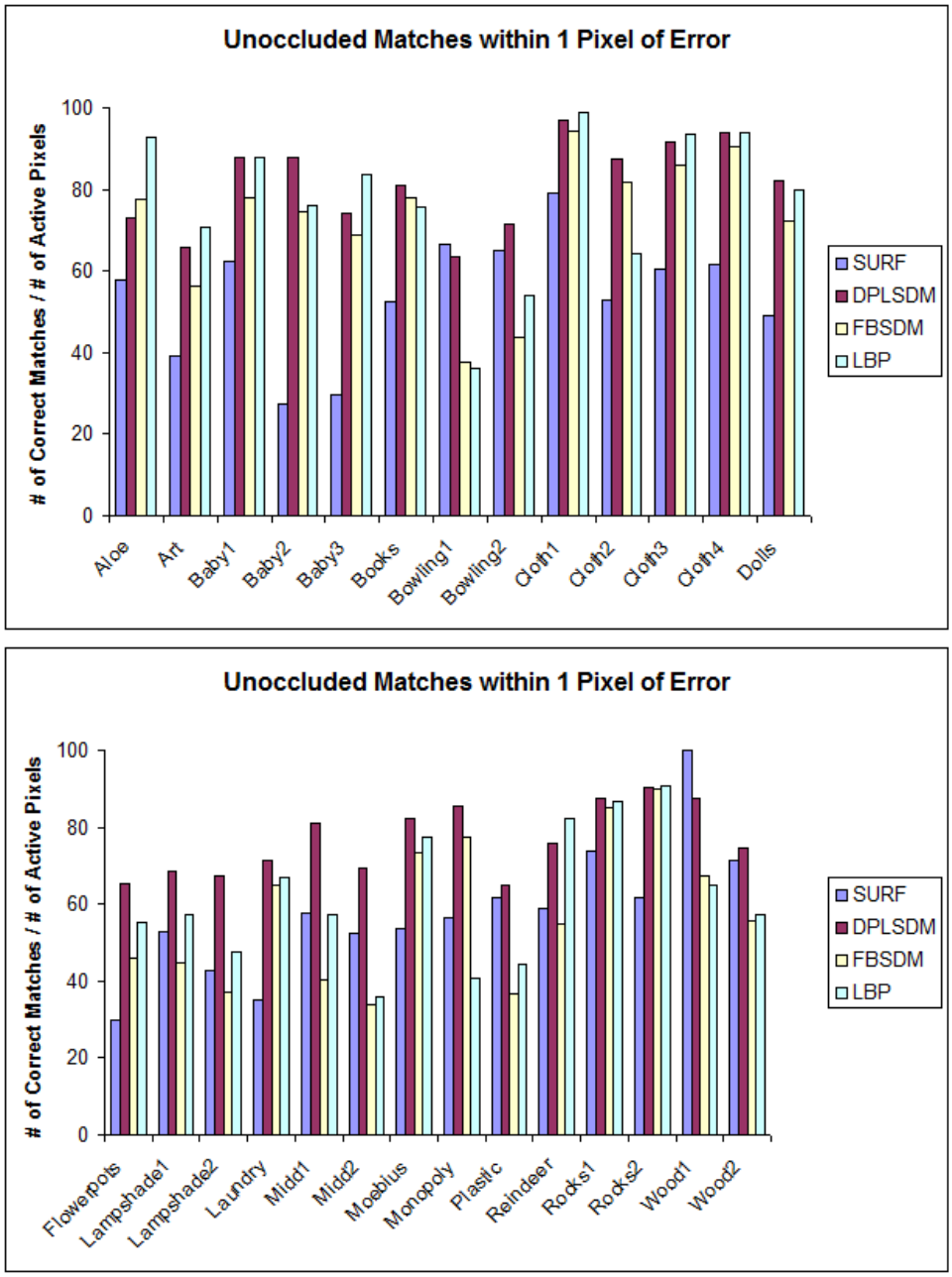

Figure 22: The matching results for unoccluded pixels within 1 pixel of error compared against the ground truth disparity image. The above graphs show the percentage of correctly matched pixels among the active pixels identified by the three feature extraction algorithms. Tests were performed using the entire 2005/2006 Middlebury Stereo set (Scharstein and Szeliski, 2002).

4.2 Quantitative Evaluation

This evaluation includes four methods of stereo matching applied to images from the 2005 and 2006 Middlebury stereo set (Scharstein and Szeliski, 2002). For each of the images in the stereo set four algorithms are used to produce a disparity image. The four algorithms

are:

1. Loopy belief propagation (LBP)
2. Speeded-up robust features (SURF)
3. Dynamic programming line segment descriptor matching (DPLSDM)
4. Flexible binary space descriptor matching (FBSDM)

LBP and SURF were selected for comparison along with DPLSDM and and FBSDM primarily due to the availablity of working implementations of the algorithms. However, they are good choices since the represent two ends of the stereo matching sectrum. SURF uses local feature points with high dimensional feature comparison, while LBP uses a global framework and simple color difference-based comparison. In addition, both methods have very few parameters that require hand tuning to achieve reasonable results.

The disparity image for each method is evaluated for the percentage of active pixels and percentage of correct matches within $+/-1$ pixel of the disparity defined by the ground truth disparity image. The $+/-1$ threshold for correctness is used to account for the sub-pixel disparity errors not handled in the matching algorithm. In addition, results are evaluated at unoccluded pixels by identifying pixels that map from the left to right and right to left disparity image. The basis for this evaluation is that if some form of occlusion detection existed in the matching algorithm, then it would discard errors in occluded regions which cannot be matched. Summary statistics are generated for each method across the entire image set, and the original per image results are also shown in Figure 22.

A paired t-test of the four methods of stereo matching is applied using the percentage of correct matches within $+/-1$ for each image set. Testing if DPLSDM > SURF produces a t-score of 7.778, DPLSDM > FBSDM produces a t-score of 6.24, and DPLSDM > LBP produces a t-score of 3.4. Comparing DPLSDM against the other three methods produces 1-sided p's less than 0.05 indicating that DPLSDM produces results that are statistically better.

Two examples where all four methods performed well are the Cloth 1 and Rocks 1 image pairs. Both of these image pairs have a smooth disparity image with very little occlusion. In addition, the objects in the environment are rich with texture; however in the Cloth 1 image set the texture is created by a repeating pattern. The SURF features performed the most poorly, and this probably occurred for several reasons. SURF will encounter problems when the features appear to be very similar, since the best match must be better than the second best match using a ratio threshold. Since the feature descriptor for each of the textured patches in the image will be very similar, it is very likely that those features will be discarded. The DPLSDM and FBSDM methods performed extremely well on the two test pairs, indicating that their feature descriptor is extremely robust when dealing with repeated textures. In general, the DPLSDM and FBSDM methods appear to perform best when the line segments or regions are small, and the disparity image is smooth. The LBP outperformed the other three methods on these two test images indicating that it achieves significant benefits when disparity images are smooth.

The Art and Lampshade 1 image pairs provided the most challenge for all four methods. These image pairs contain highly variable disparity images with large areas of occlusion. In addition, they contain a number of thin objects that occlude textured surfaces.

As a result, the descriptor generated at the boundary between the thin object and the textured surface can be very different. Since SURF, DPLSDM, and FBSDM use boundary descriptors they are all degraded by the complex boundary interaction. A problem specific to the FBSDM method is visible in these image pairs as well. When large areas of untextured surfaces are extracted it is very likely that they will be cut into multiple pieces due to the splitting criteria. If a region contains only points inside the untextured surface, then the region will match to some arbitrary location inside the untextured surface, causing all matches within that region to be incorrect. The LBP was degraded significantly as well, which is most likely caused by the sharp changes in the disparity image. The sharp changes prevent the smoothness term from having a strong effect on the outcome, therefore data matching is more heavily relied on.

4.3 Qualitative Evaluation

To aid in evaluation, the Middlebury stereo set are preprocessed to apply constraints to the stereo pairs. All image pairs in the Middlebury set are aligned using epipolar rectified, so the search space is limited to the same row in both the left and right image. Also, the minimum and maximum horizontal disparity is known for all stereo pairs. Although these constraints are used in the evaluation of the SURF, DPLSDM, and FBSDM stereo methods, it will be shown that all three methods work well even when these constraints are not maintained. The LBP has been excluded from these tests due to a lack of support for 2-D search spaces and an extremely large memory requirement.

The problem with constrained image pairs is that they represent a best case environment for solving stereo vision problems. In mobile robotics, the environment is far more dynamic, with many unknowns with regards to the camera position and capture settings. To rectify image pairs camera calibration or an initial estimate of the epipolar geometry must be performed. The camera calibration can be represented in terms of absolute camera positions with the essential matrix, or using the estimates in the fundamental matrix. The calibrations can be precomputed and used very effectively if the two cameras are locked in position relative to one another other. A problem arises, however, if the robot must change the camera's convergence distance to adjust focus between near and far objects. Changing the convergence distance of the two cameras changes the configuration of the stereo cameras, a property that is difficult to compensate for using precomputed calibrations. Allowing variable convergence also introduces a problem with minimum and maximum disparity ranges, making it difficult to constrain the search space. In addition, unrectified image pairs have both horizontal and vertical disparity components that increase the search from a 1-D line to a 2-D region. Since no ground truth disparity image exists for the natural scenes discussed in this section, a qualitative comparison will be used.

The similarity of displacement images is simply evaluated by finding an assigned displacement in the SURF output and locating a nearby measurement in the same position of the DPLSDM and FBSDM output. If the displacement in all three methods are within a few pixels they are considered similar. Due to the difference in the location of assigned feature point, accurate quantitative result are difficult to extract from the output. As a result, visual inspection of the similarity between the output is suggested.

The hand scene (see figure 23) is captured using two average quality web cameras that

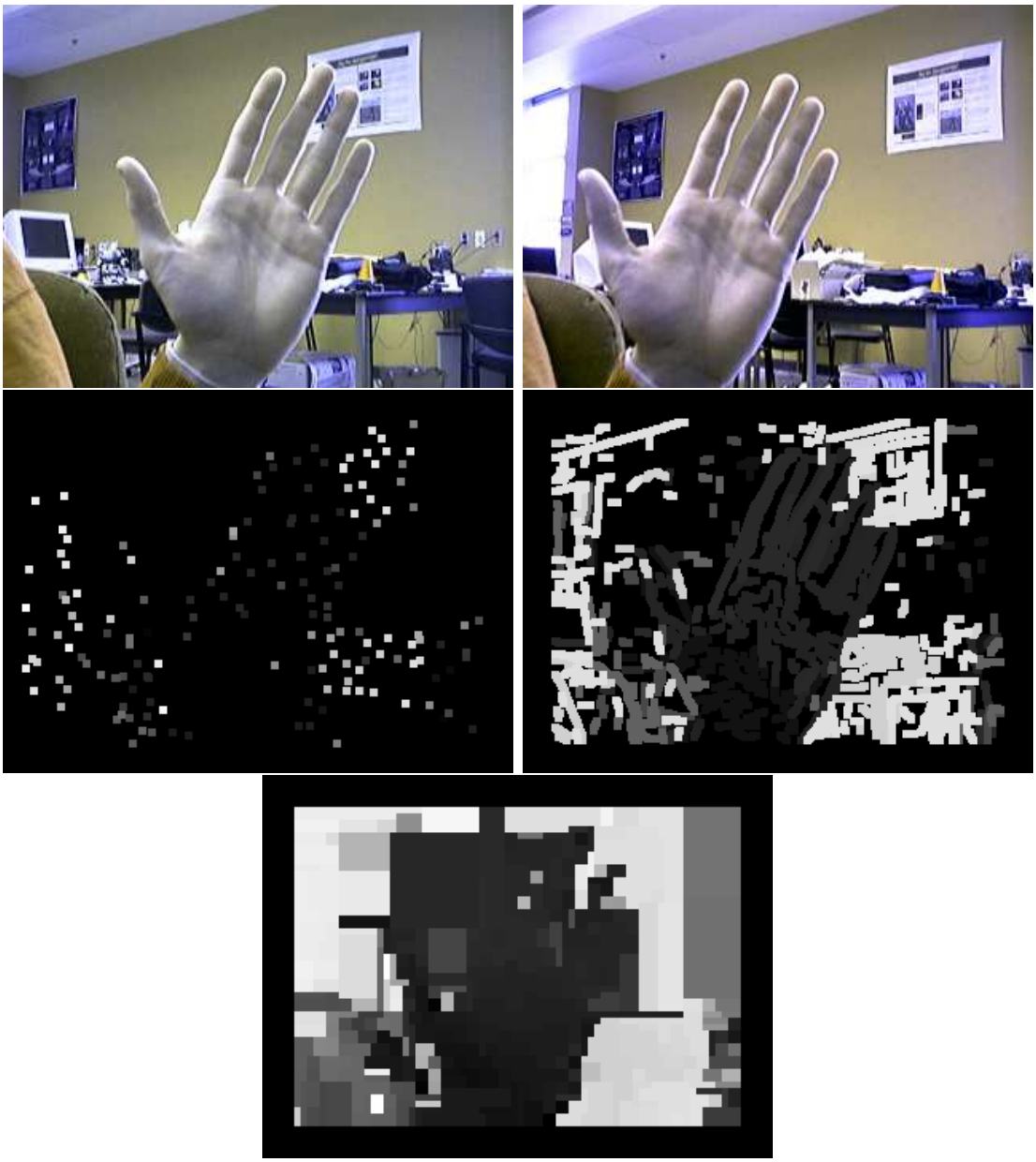

Figure 23: Hand. Images of a hand against an indoor background. The top left and right images are the original captured images. The resulting disparity image from the left image for the SURF (middle-left), DPLSDM (middle-right), and FBSDM (bottom) based stereo. Dark pixels are near to the camera, bright pixels are further from the camera.

produce low levels of noise and have similar colour properties. Prominent features in the scene include a hand near the camera, desks in the background, a solid coloured wall, and high saturation in one image from a window. This scene has a large disparity range in both the horizontal and vertical direction. For the SURF, DPLSDM, and FBSDM stereo methods a search space of $+/-40$ pixels horizontally and $+/-20$ pixels vertically is used on the 320x240 pixel images. The resulting disparity images appear very similar across the three methods. The region-based method produces a large area of error in the

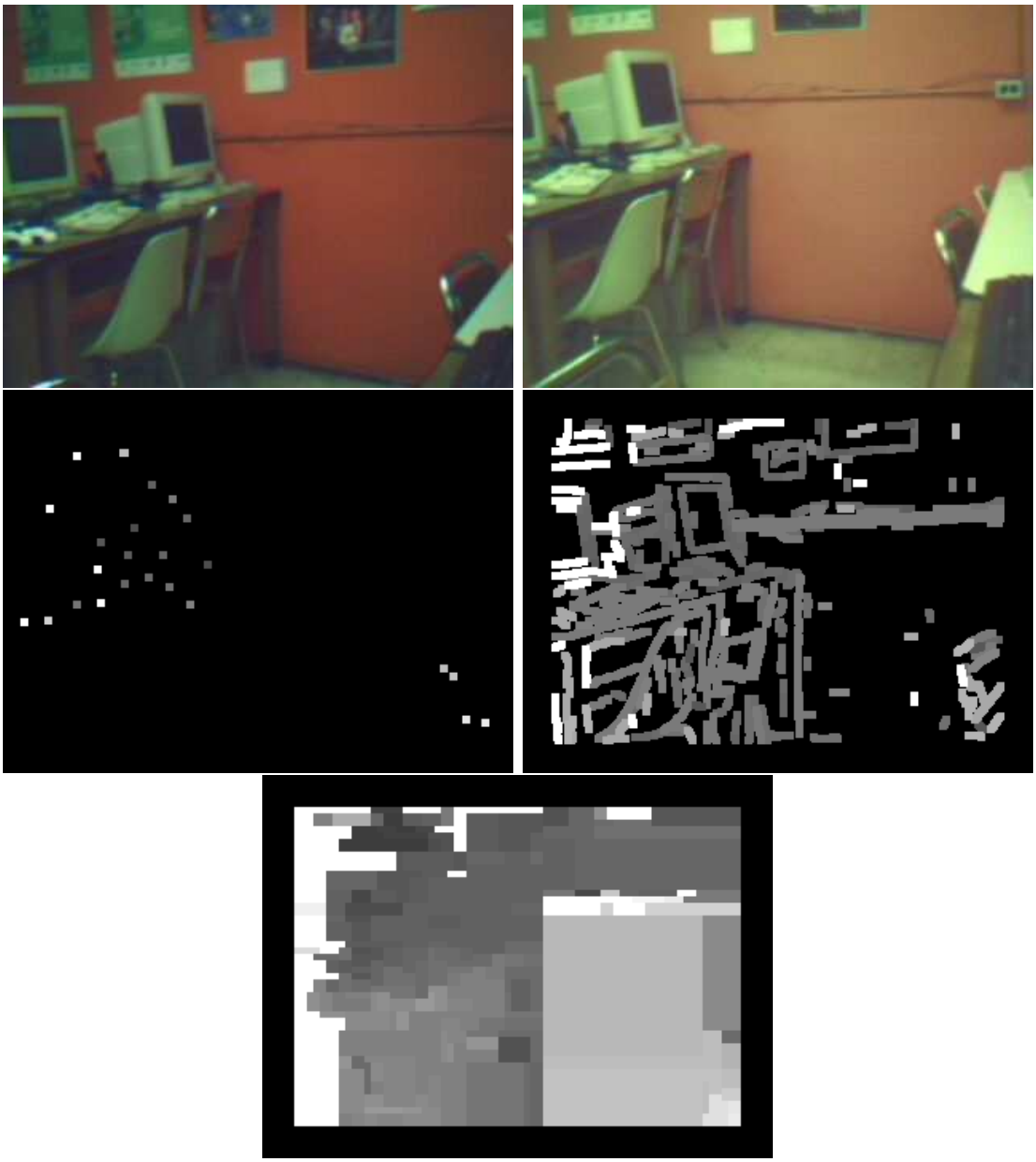

Figure 24: The Dungeon. Images of an indoor scene with a large disparity and colour difference between the two cameras. The top left and right images are the original captured images. The resulting disparity image from the left image for the SURF (middle-left), line segment (middle-right), and region (bottom) based stereo matching. Bright pixels are near to the camera, dark pixels are further from the camera.

untextured area of the image. All three features produce incorrect results in the occluded area to the left of the hand, as expected due to a lack of occlusion handling.

The dungeon scene (see figure 24) is captured using two poor quality web cameras. The colour difference between the two frames is significant due to brightness/contrast differences in the camera capture settings. Like the previous example, this scene has a very large horizontal and vertical disparity. The search space used for matching in this

image set is $+/-40$ pixels horizontally and $+/-40$ pixels vertically with an image size of 320x240. Once again all three methods produce similar disparities, with the region based method failing on the large untextured area of the wall. This example demonstrates that all three methods are extremely robust to colour differences in the capture device due to the fact that they primarily focus on colour gradients.

4.4 Real-time Performance Evaluation

Runtime is an important consideration when choosing a stereo algorithm for real-time tasks including USAR robotics. LBP, SURF, DPLSDM, and FBSDM are evaulated using a Linux operating system with a single core 1.8Ghz processor with 1GB of RAM. The performance evaluations are done with the Middlebury test images for consistency. For each image pair, the loopy belief propagation, SURF, line segment and region-based stereo algorithms are run five times with mean and standard deviation shown in seconds of processing time in figure 25.

The first difference worth noting in the runtimes is that the LBP requires significantly more time than the feature-based methods. High runtimes for the LBP result from every label in every pixel requiring processing on each iteration. Feature based methods generally only require a few passes over every pixel in the input image before the data set is reduced for matching. Figure 25 clearly demonstrates the cost of using a global solution for the stereo matching problem.

The SURF, DPLSDM, and FBSDM stereo methods achieve similar runtime performance which is generally below one second per frame as shown in figure 26. The books image pair is a good example of when the environment is computationally expensive for the DPLSDM. When the DP table used in the line segment matching involves many long line segments the algorithm can require a significant amount of processing. In general, the runtimes for these three methods are similar enough to not have a significant impact on the choice of stereo algorithms.

5 Conclusion

This paper provides a description and comparison of four methods for stereoscopic disparity extraction. The evaluation shows that the DPLSDM provide more reliable results then LBP, SURF, and FBSDM for each matched pixel. Evaluations of accuracy using the 2005 and 2006 Middlebury stereo set show that on average DPLSDM outperformed LBP by 10.8%, FBSDM by 14.2%, and SURF by 22.8%. DPLSDM also provides runtime performance similar to SURF and FBSDM, while significantly outperforming the LBP's global framework. Though DPLSDM produces more accurate results using the defined accuracy metric it is still semi-dense with an average of 9.6% coverage in the disparity image. In addition it provides no framework for infilling untextured regions or identifying occluded regions of the image.

The main contribution of this paper is the DPLSDM and FBSDM algorithms that have been described and evaluated in this paper. The development of DPLSDM included the introduction of the EC data structure for storing gradient orientations. The EC data structure provides a computationally efficient method of storing and performing operations on orientation information. The introduction of this data structure was essential

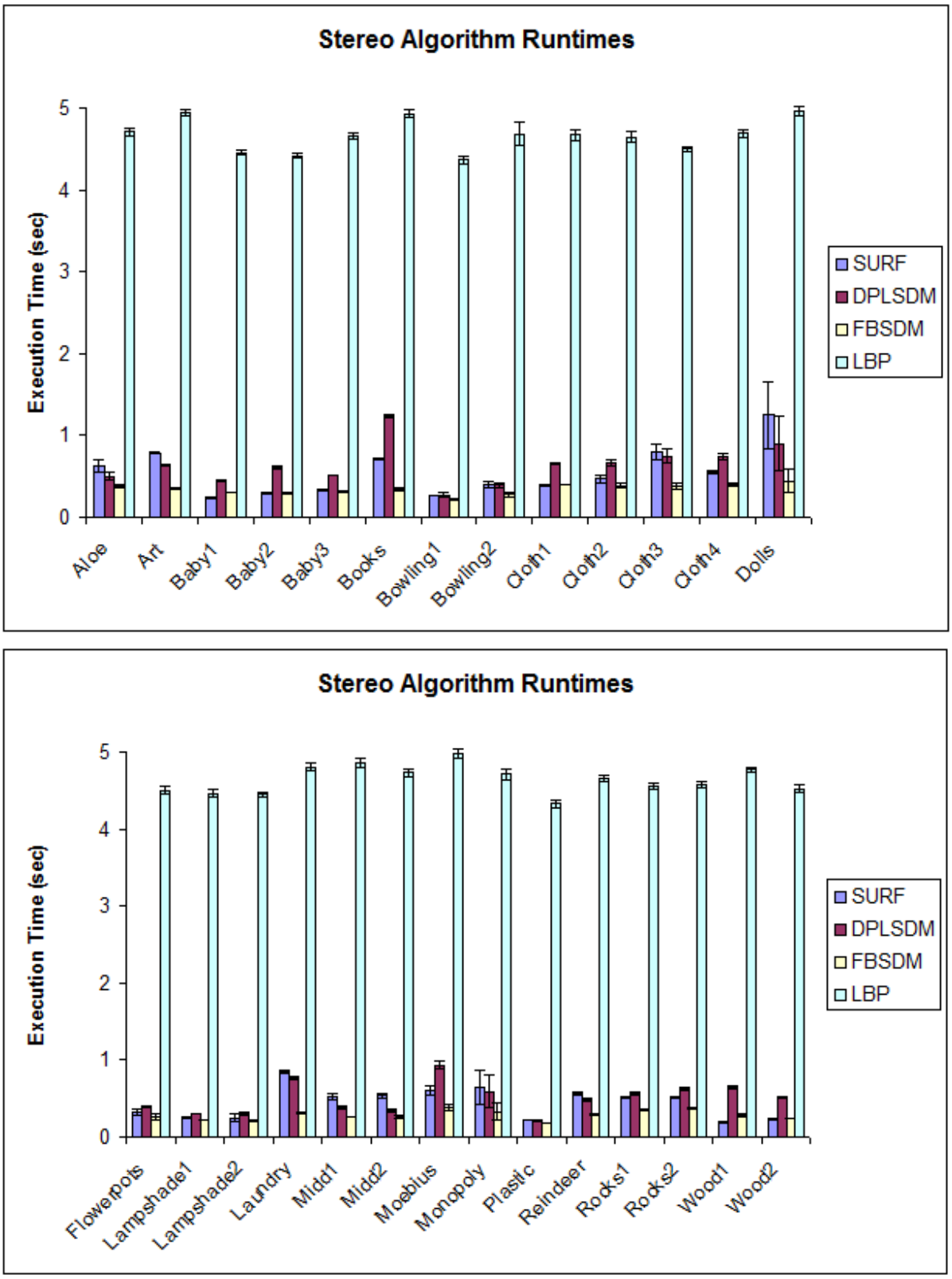

Figure 25: The runtime statistics of the four stereo algorithms used in the quantitative evaluation. Tests were performed using the entire 2005/2006 Middlebury Stereo set (Scharstein and Szeliski, 2002).

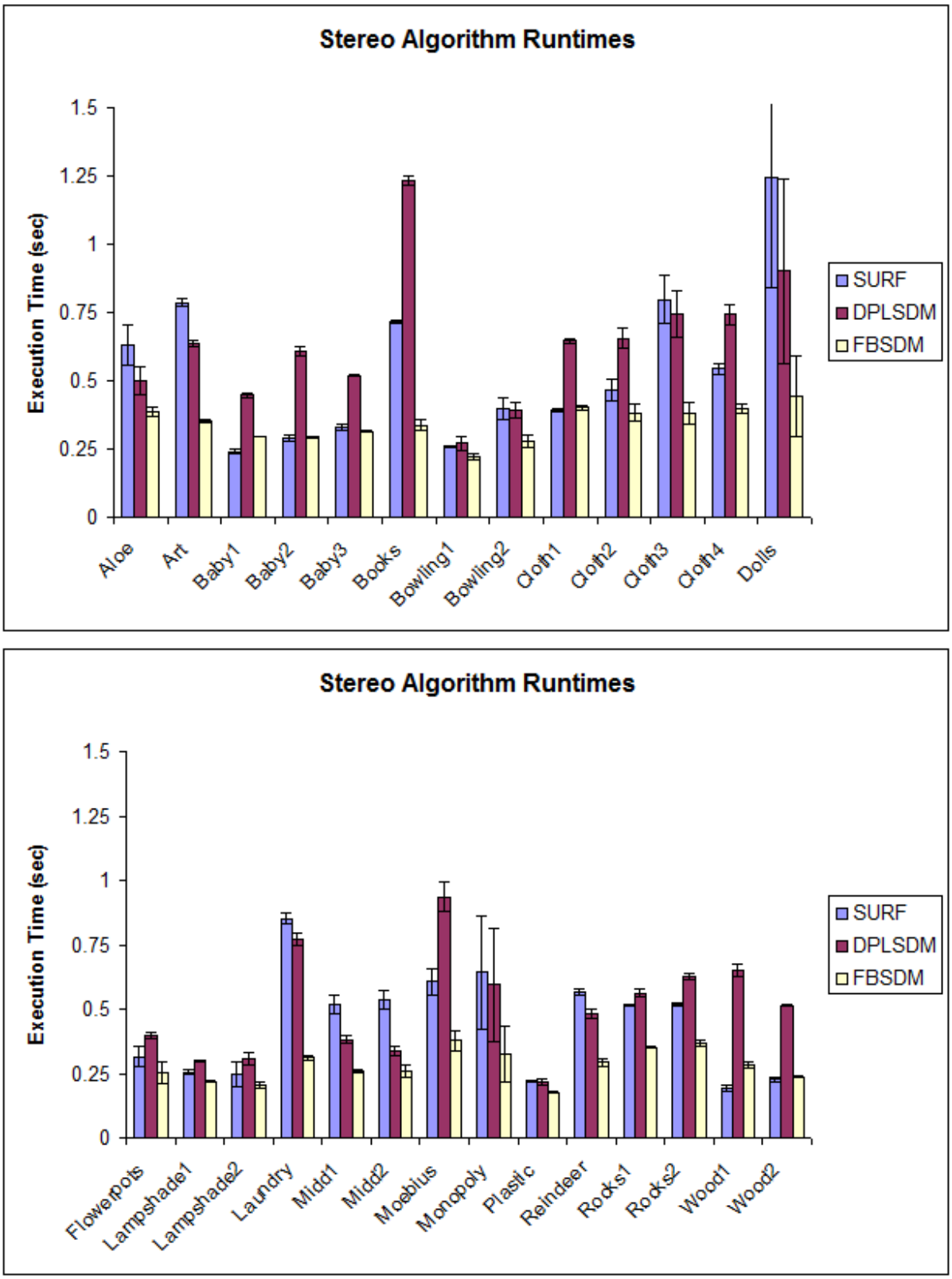

Figure 26: The runtime statistics of the three local feature based stereo algorithms used in the quantitative evaluation. Tests were performed using the entire 2005/2006 Middlebury Stereo set (Scharstein and Szeliski, 2002).

to improving the performance of the gradient thinning and line segment extraction algorithms.

The line segment extraction algorithm allows fast local extraction of line segment features from an image. Combined with the filtering stage, the algorithm produces high quality line segments for use in stereo matching. With some additional effort focused on infilling gaps produced by noise in the image, I believe this method of extraction could produce some of the highest quality line segments available for computer vision applications.

The line segments and feature descriptors extracted at each point provide the frame work for a dynamic programming solution which achieves high levels of accuracy. The key to making use of dynamic programming in this algorithm was the replacement of the standard minimization and penalty value with a maximization and count table. The maximization and count produce accurate results even when matching different size line segments, without requiring a hand tuned penalty value. The use of dynamic programming to solve line segment matching is quite novel, however, I believe that this matching method can be significantly improved upon as discussed in the future work section.

The FBSDM introduces a new method of performing fast region segmentation. The fast segmentation uses flexible binary space partitioning and integral images to produce a fast over-segmented image. The regions produce good-quality dense disparity images with the exception of regions in large open spaces. The FBSDM could be improved upon by adding a post segmentation merging stage that combines neighbouring regions if there is no active pixels seperating them, but this is left for future consideration.

In summary, the DPLSDM and FBSDM explore an interesting new approach to generating stereo displarity images. With some refinement, I believe these data structures and algorithms could provide a significant improvement to stereo vision systems and computer vision in general.

5.1 Future Work

A limitation of the DPLSDM implementation is that errors early in the dynamic programming table propagate through the rest of the solution. Future work should focus on replacing the dynamic programming structure with a more robust method of sharing information between nodes on the line segment. One option would be to use belief propagation with information shared between neighbouring points on the line segment. Unlike the dynamic programming solution, belief propagation would move information both forwards and backward through the line segment. In addition, since the line segment does not contain loops it is guaranteed to converge to a solution.

The feature descriptor is also a useful area for further study. Since the feature descriptor is fundamental to any stereo matching algorithm it should be a priority to find the best possible method of extracting the descriptor. A good feature descriptor must balance the speed and quality of the extraction and matching process. This paper demonstrates that integral images provide a fast method of feature extraction, but the feature is limited to being built with box filters. Further investigation will be necessary to determine if the speed benefits of integral image features offset the loss in precision of the box filter.

References

Stefania Ardizzoni, Ilaria Bartolini, and Marco Patella. Windsurf: Region-based image retrieval using wavelets. In *DEXA Workshop*, pages 167–173, 1999. URL `citeseer.ist.psu.edu/ardizzoni99windsurf.html`.

Jacky Baltes and John Anderson. Flexible binary space partitioning for robotic rescue. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3144–3149, Las Vegas, October 2003.

Herbert Bay, Vittorio Ferrari, and Luc Van Gool. Wide-baseline stereo matching with line segments. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 329–336, 2005.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*, May 2006.

Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *In Proc. IEEE Conf. Comp. Vision Patt. Recog*, pages 1000–1006, 1997.

Michael Bleyer and Margrit Gelautz. A layered stereo matching algorithm using image segmentation and global visibility constraints. *ISPRS Journal of Photogrammetry and Remote Sensing*, 59(3):128–150, 2005.

Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via cut graphs. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

J. Canny. A variational approach to edge detection. *AAAI-83*, 1983.

Thomas H. Corman, Charles E. Leiserson, Ronald L. Rivest, and Cliffort Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Ma, 2nd edition, 2001.

Olivier Faugeras and Quang-Tuan Luong. *The geometry of multiple images*. MIT Press, Cambridge, Ma, 2001.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. In *In CVPR*, pages 261–268, 2004.

Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. ISBN 0201180758.

Madasu Hanmandlu, Vamsi Krishna Madasu, and Shantaram Vasikarla. A fuzzy approach to texture segmentation. In *Proceedings of the International Conference on Information Technology: Coding and Computing*, pages 636–642, 2004.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. ISBN 0521540518.

M. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1338–1359, December 1997.

Nicholas J. Hollinghurst. *Uncalibrated stereo hand-eye coordination*. PhD thesis, University of Cambridge, January 1997.

Jeong-Hun Jang and Ki-Sang Hong. Fast line segment grouping method for finding globally more favorable line segments. *Pattern Recognition*, 35(10):2235–2247, 2002.

Nezamoddin N. Kachouie, Javad Alirezaie, and Paul Fieguth. A hybrid algorithm using discrete cosine transformation and gabor filter bank for texture segmentation. In *Canadian Conference on Electrical and Computer Engineering*, pages 1805–1808, 2004.

Euijin Kim, Miki Haseyama, and Hideo Kitajima. Fast line extraction from digital images using line segments. *Systems and Computers in Japan*, 34(10), 2003.

David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999. URL `citeseer.ist.psu.edu/lowe99object.html`.

Bruce D. Lucas and Takeo Kanada. An iterative image registration technique with an application in stereo vision. In *Proceedings DARPA Image Understanding Workshop*, pages 121–130, 1981.

Brian McKinnon and Jacky Baltes. Practical region-based matching for stereo vision. In Reinhard Klette and Jovisa D. Zunic, editors, *IWCIA*, volume 3322 of *Lecture Notes in Computer Science*, pages 726–738. Springer, 2004. ISBN 3-540-23942-1.

Brian McKinnon, Jacky Baltes, and John Anderson. A region-based approach to stereo matching for usar. In Ansgar Bredenfeld, Adam Jacoff, Itsuki Noda, and Yasutake Takahashi, editors, *Proceedings of RoboCup-2005: Robot Soccer World Cup IX*, Osaka, 2005.

M. Mirmehdi, P. L. Palmer, and J. Kittler. Robust line-segment extraction using genetic algorithms. In *6th IEE International Conference on Image Processing and Its Applications*, pages 141–145. IEE Publications, 1997. ISBN 0 85296 692 X. URL `citeseer.ist.psu.edu/284018.html`.

Don Murray and Cullen Jennings. Stereo vision based mapping and navigation for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1694–1699, 1997.

Gou-Chol Pok, Jyk-Charn Liu, and Keun Ho Ryu. Fast estimation of the number of texture segments using co-occurrence statistics. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 341–344, 2004.

V. Salari and Ishwar K. Sethi. Feature point correspondence in the presence of occlusion. *IEEE Transactions on Pattern Analysis and Machine Learning*, 12:87–91, 1990.

D. Scharstein and C. Pal. Learning conditional random fields for stereo. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, June 2007.

D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47((1/2/3)):7–42, April-June 2002.

D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, 1:195–202, June 2003.

Stephen Se, David Lowe, and Jim Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21(3):364–375, June 2005.

Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

Masaaki Shibata and Hajime Kawasumi. solution for stereo correspondences on active stereo vision robot. In *AMC*, pages 665–670, 2004.

Harris Sunyoto, Wannes van der Mark, and Dariu M. Gavrila. A comparative study of fast dense stereo vision algorithms. In *IEEE Intelligent Vehicle Symposium*, pages 319–324, 2004.

P. Tay, J. P. Havlicek, and V. DeBrunner. Discrete wavelet transform with optimal joint localization for determining the number of image texture segments. In *International Conference on Image Processing*, pages 281–284, 2002.

Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, March 1998.

Christopher Urmson, M Bernardine Dias, and Reid Simmons. Stereo vision based navigation for sun-synchronous exploration. In *IROS 2002*, 2002.

Iris Vanhamel, Ioannis Pratikakis, and Hichem Sahli. Multiscale gradient watersheds of color images. *IEEE Transactions on Image Processing*, 12(6), June 2003.

Olga Veksler. Semi-dense stereo correspondence with dense features. In *IEEE Computer Vision and Pattern Recognition*, December 2001.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. *Exploring artificial intelligence in the new millennium*, pages 239–269, 2003.

Zhengyou Zhang. Estimating motion and structure from correspondences of line segments between two perspective images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17 (12):1129–1139, 1995.

Hongwei Zhu and Otman Basir. Proximity measure image based region merging for texture segmentation through gabor filtering and watershed transform. In *Proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, pages 742–747, 2003.