Keystone Rescue Techniques

Shawn Schärer
Department of Electrical and Computer Engineering
schaerer@ee.umanitoba.ca

Brian McKinnon and **Jacky Baltes**
ummcki01@cc.umanitoba.ca jacky@cs.umanitoba.ca

John Anderson and **Ryan Wegner**
andersj@cs.umanitoba.ca wegner@cs.umanitoba.ca
Department of Computer Science
University of Manitoba
Winnipeg, Canada

Abstract

This paper will describe the current research and technology that is implemented the Keystone rescue team.

Introduction

This year will be the third year that we have competed at the AAAI USAR competition. Our research focus is on developing fully autonomous robots that can maneuver in unstructured environments. All of the work that is being done by our group is computer vision based, utilizing low level image processing (edges, skeletons, regions, etc...) to perform ego-motion estimation, stereo matching and VSLAM (Visual simultaneous localization and mapping).

In order to keep our research purely in the computer vision domain, we use cameras as the sole sensor (e.g., no LADAR, shaft encoders, etc...). This allows us to develop robust algorithms using inexpensive (noisy) platforms.

Section describes the robots used in the research. Section provides background information on previously approaches to stereo vision, ego-motion and SLAM . Section describes the design and implement of our system . Section explains our approach to ego-motion estimation and section will detail our stereo vision work. A brief discussion of the results as well as directions for future research are in Sec.

Robot platforms

This section describes the hardware of our robots. As mentioned previously, one of our main goals is the development of robust systems. Furthermore, we believe that through the use of cheap hardware with noisy sensors and inaccurate accuators requires fundamentally more robust systems than approaches that rely on highly accurate sensors and actuators.

So, similar to our previous robot designs, the 2004 Keystone Rescue team uses cheap components. However, based on our experiences in previous years' competition, we developed two new platforms: (a) ZaurusBot and (b) Spike.

The ZaurusBot is a fully autonomous approach based on a commonly available Sharp Zaurus PDA. Since the problem of finding victims and generating a map using visual feedback alone is a very difficult one, we decided this year to also include a tele-operated robot, Spike, in the competition.

ZaurusBot

The ZaurusBot is a simple design that is loosely inspired from a scorpion. The chassis is constructed out of aluminum and plastic. The main electrical and computation components that are used are two modified standard RC servos that act as left and right motor of a differential drive setup, a CMU cerebellum microcontroller for servo control/host communication and a Sharp Zaurus PDA. The PDA provides powerful processing for complex computer vision and mapping algorithms.

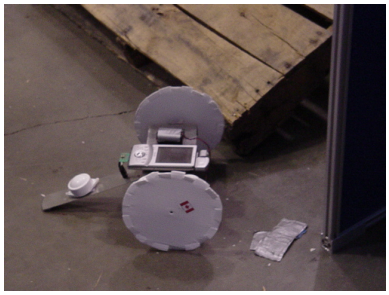

Figure 1: ZaurusBot platform in the red arena

Spike

Spike is our first low cost stereo vision system. The chassis is based on an off the shelf 1:6 scale RC PT Cruiser and uses a 533MHz Via Mini-itx board for processing. Two USB web cameras provide vision input. To increase the versatility of the system, the two cameras are mounted on a pan unit, which is controlled by a RC servo. Spike uses a Linux kernel

with hard real-time extensions. This allows us to control the pan servo as well as the control of the drive motors without additional hardware.

Figure 2: Spike in the red arena

Although these two robotic platforms seem simple, they provide robust vehicles for deploying a mobile platform with vision in the yellow arena. ZaurusBot and Spike also provide sufficient processing power for on-board vision processing.

Currently, the main focus of our research is to estimate the motion of robots with very poor odometry using the change in vision sequences from on-board cameras. The idea is that by watching how a single image (in the case of the ZaurusBot) or from a stereo image pair (in case of Spike) changes, one can deduce the motion of the robot. Estimating a robot's motion presents a first step. This information can then be used to create maps and localize the robot within an unstructured environment.

Related Work

Ego-motion estimation, the problem of determining the motion of a robot from a given a sequence of images taken during the motion, is a very popular research area in computer vision. Detecting motion across a series of images is a useful task in many applications beyond its use in ego-motion detection, from manufacturing to security, for example. While this is easy for a human to do, it is a difficult problem to approach from the standpoint of a computer program.

More formally, ego-motion estimation can be defined as determining the translation and rotation parameters of a camera view given two different views of a scene. This estimate is usually based on unconstrained motion of a calibrated camera with respect to a plane. In this case, there are eight parameters to be determined.

Most approaches use the *rigid world* assumption where the scene is considered static and all movement in the image is due to the motion of the robot. In multi-agent domains such as robotic soccer, this assumption does not hold true very often. For a local vision robot, large movements in the image may be due to other robots moving in the image as well.

Previous work in ego-motion estimation in computer vision can be divided into two categories: those that presuppose structuring (e.g. lines) in the image to use as a basis for detecting movement, and those that do not. As an example of the latter method, Stein et al propose a method that takes a region of an image, and uses all pixels in the region to provide support for a particular motion (Stein, Mano, & Shashua 2000). Their method computes a global probability distribution function for the whole image, and is robust in the presence of many outliers. Stein reduces the number of parameters to three: translation, pitch, and yaw.

As an example of a method attempting to exploit structure in the image, Zhang (Zhang 1995)describes a methodology for estimating motion and structure using line segments in an image . This approach is based on the assumption that line segments overlap in 3D space, which allows a reduction in the number of motion parameters.

Szeliski andTorr (Richard Szeliski)introduce geometric constraints early in the structure reconstruction phase to improve the overall 3D reconstruction results. This method takes advantage of the fact that, the real world has many planer structures such as vertical walls and flat ground planes and uses external geometric constraints such as plane parallelism to reduce reconstruction errors.

Sim and Dudek (Robert Sim 1998) utilize learned landmarks to estimate a robot's position. Their method first attempts to visually detect possible landmarks by looking for image features and then matches these detected landmarks against a predefined landmark database. The robot's position can be determined if sufficient landmarks are found and matched.

Another example of exploiting structure for an image is from Se, Lowe and Little (Stephen Se) who developed a local vision based SLAM algorithm by following visual scale invariant feature transform (SIFT) landmarks in an unstructured environment. To estimate the ego-motion of the robot they match SIFT features between frames and perform a least squares minimization of the match errors. This yields a better 6 DOF ego-motion and localization estimate than previous approaches.

Approaches in both categories have strong theoretical foundations, but there are also practical issues to be considered when deploying these methods on real mobile robots. In spite of today's embedded systems becoming more powerful, they are still much more limited than the workstations on which many of these approaches have been implemented and evaluated. Consequently, most existing approaches are computationally too expensive to be implemented on an embedded controller with real-time constraints.

The approach described in this paper uses a two parameter model appropriate for today's embedded systems. Given a fast camera calibration routine and a fast wall detection method, we are able to estimate the motion using only a cheap, common, off-the shelf embedded system. The model used in our work is described in section , followed by the two specific methodologies we combine in our implementation.

System Design

Figure 3 shows a block diagram of our system. The mobile robot interacts with the environment using actuators and re-

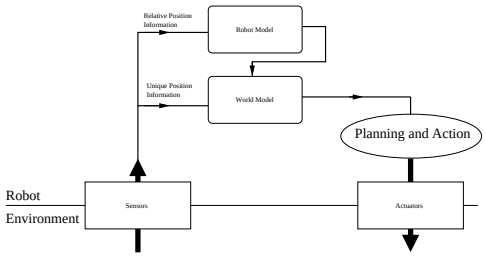

Figure 3: System Design of our Localization Method

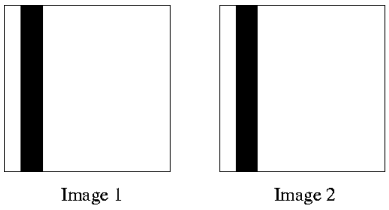

Figure 4: A simple example where it is impossible to determine the angular, but not the linear velocity of a mobile robot. Since the relative orientation and position of the line has not changed, the robot was moving in a straight line.

ceives feedback from the environment via its sensors — in our case, visual feedback from a CMOS camera.

The sensors are used to update the world model of the robot — in our case, its orientation and position. However, not all sensor feedback is useful to update the world model uniquely. For example, assume that a robot tracks parallel to a line as shown in Fig. 4. Given the image sequence, it is clear that the mobile robot has traveled in a straight line, but it is not possible to determine how far it has traveled. Therefore, only information about the angular velocity and position can be determined uniquely.

Most other systems use dead reckoning to supplement the localization in this situation. In our case, we wish to avoid relying upon methods that can only be assumed to be correct under restricted conditions (such as the use of shaft encoders). As an alternative, our system maintains an internal model of the robot to support dead-reckoning style navigation.

In this approach, sensor feedback is used to update an internal robot model, which models the behavior of the robot to the motor commands. This relationship changes drastically over time: for example, since the batteries begin to lose their charge after only a few minutes, the robot moves a much shorter distance per unit time with a given motor setting as time passes. The robot model allows the system to consider the current reactions of the robot to different motion commands, and is adapted when new location information becomes available.

As stated in Section , our approach to ego-motion estimation is based on two coordinating methods: one that does not assume any structure in the image, and the other that does. We will describe these two methods in the following

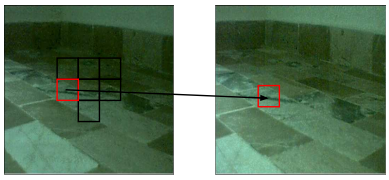

Figure 5: Ego Motion Estimation Using Optical Flow

sections.

Ego-motion estimation

Ego-motion estimation is a interesting problem in robotics. Current techniques of estimating a robots self motion (Eg shaft encoders) have a problem of producing inaccurate data in situations where wheel slip is predominate (Eg rubble piles, sand / gravel surfaces). We are purposing two technique that would be immune to such problems and that gives accurate ego-motion estimation or can be fused with other sensors to provide a highly accurate and robust ego-motion estimation sensor toolkit. The first approach uses the optical flow of small patches in the view of the camera to determine translation and rotation from the previous to the current view. This approach is general in nature and does not rely on any specific features of the image itself. The second approach uses real or virtual lines to determine translation and rotation by measuring how the robot moves in relation to lines detected in the image. Since these two approaches rely on different information in the image, they are complimentary.

Ego Motion Estimation Using Lines

This method as described in the previous section is general-purpose, in that it does not assume any specific structuring of the scene in the images. However, there is information that can be obtained from a scene that can be used to supplement the performance of a general method such as this one. One of the most obvious and striking features that appears in a wide variety of environments are lines. For example, the edges between a wall and the floor form lines in an image, as do the edges between walls themselves. While in many cases walls are the most predominant source of lines, lines also exist in terms of patterns on floors or walls (such as the many lines shown in Figure 5).

Lines are thus a natural choice for a reasonably general feature to attempt to take advantage of in order to improve general-purpose ego-motion detection approaches such as those presented above.

We explain this approach using the kinematics of a differential drive robot (summarized in Figure 6). Our approach depends on the assumption that the velocities of the right and the left wheel are constant during the time period for which the velocities are to be estimated. This is necessary since the approach uses the differences between the start and end locations of the robot motions to derive values for the wheel velocities. This is impossible if the wheel velocities are not constant during the time period in question. If this restric-

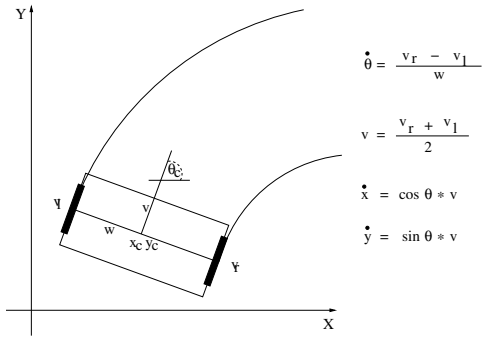

Figure 6: Kinematic Model of a Differential Drive Robot

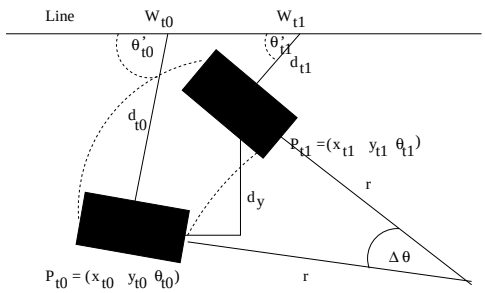

Figure 7: Kinematic Model of a Differential Drive Robot

tion was not made, the robot could perform any one of an infinite number of possible movements and return to the end location.

In order to motivate the analysis, consider the example depicted in Figure 7. At time $t0$, the robot is at position $P_{t0} = (x_{t0}, y_{t0}, \theta_{t0})$, and a straight line (formed by a wall) is in front of the robot. Let W_{t_i} be the point on the wall in the center of the camera view of the robot at time t_i and let d_{t_i} be the distance between P_{t_i} and W_{t_i}. Let θ'_{t_i} be the angle between the orientation of the robot θ_{t_i} and the angle of the wall θ_w.

At time $t1$, the robot is at position $P_{t1} = (x_{t1}, y_{t1}, \theta_{t1})$. The distance d of the robot to the wall as well as the angle θ' between the robot and the wall will have changed.

The problem is to derive from this information the current wheel velocities v_r and v_l for the right and left wheel respectively.

Given the kinematic model of the differential drive robot (Figure 6), it is easy to see that:

$$\dot{\theta}' = \dot{\theta} = \frac{v_r - v_l}{w}$$

Since the width w of the robot is known, this allows us to compute the difference in velocities for the two wheels. However, there is not sufficient information in the turn rate alone to determine the average velocity.

We assign a coordinate system X' with the origin at P and the x-axis parallel to the line or wall. In this coordinate system, the robot is at the origin and its orientation is θ'.

We can then derive the distance from the robot to the closest point on the wall $d_W = \sin\theta' * d$, allowing us to compute:

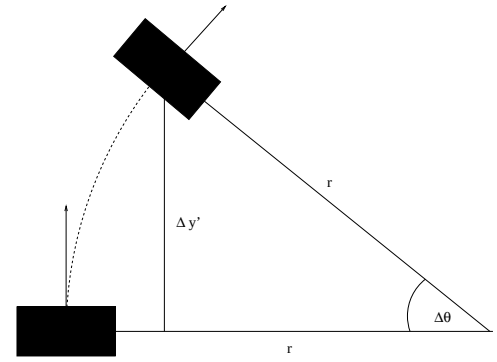

Figure 8: Determination of the linear velocity v of a differential drive robot facing a wall

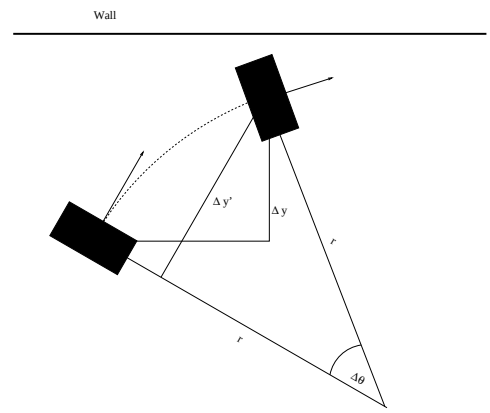

Figure 9: Determination of the linear velocity v of a differential drive robot with an arbitrary angle to the wall

$$\Delta y = -(d_W(t1) - d_W(t0))$$

As can seen in Figure 8, the linear velocity of the robot can be computed from its rate of turn and the offset in the direction of the robot's $\Delta y'$.

From a simple geometric relationship, we can compute the radius of the circle r:

$$r = \frac{\Delta y'}{\sin(\theta'_2 - \theta'_1}$$

The linear velocity is then equal to the length of the circle segment $(r * \Delta\theta)$ divided by the time to cover this distance:

$$v = \frac{r * (\theta'_2 - \theta'_1)}{\Delta t}$$

In case the robot is facing the wall directly (i.e., $\theta' = 90^o$), and the above equation yields the correct solution for the linear velocity of the robot.

As can be seen in Figure 9, similar reasoning allows us to compute the circle distance $\Delta y'$ from a given wall distance.

From the figure, one can derive that:

$$\frac{c}{dy} = \sin\theta'_1 - \frac{\theta'_2 - \theta_1}{2}$$

In this case:

$$\Delta y = \Delta y' * \frac{\sin\frac{\theta_2-\theta_1}{2}}{\cos\frac{\theta_2+\theta_1}{2}}$$

The right and left wheel velocities can then be calculated as:

$$v_r = \frac{2v + \theta' * w}{2} v_l = \frac{2v - \theta' * w}{2}$$

As stated earlier, any straight line segment in the image can be used as a guideline, allowing this method to be broadly applicable. While source of lines were hypothesized as walls in a typical indoor environment in the analysis above, lines are present in many important environments. In robotic soccer (and indeed, most human sports) lines are an important element of play. Even in less structured examples, lines are still present. In robotic rescue environments, for example, even though most of a structure may be collapsed, lines still exist in any remnant of standing wall, in debris with straight edges (e.g. strewn papers, lumber or other structural components), and in decorative patterns used in indoor environments. All of these provide line segments that can be used to compute the incremental location of the robot.

Implementation

The following algorithm description goes into detail about practical aspects of the ego-motion estimation on the ZaurusBot.

First, Sobel edge detection is applied to the raw input image. Strong edge pixels are extracted from the image through thresholding. Strong lines in the image are extracted by using a Hough transform. To speed up the Hough transform, we only look for lines with orientations that allow us to distinguish changes in a line's orientation and distance to the robot. In our case, this means we are limiting ourselves to approximately horizontal lines (70 to 110 degrees). We also extract vertical lines since they occur often and allow us to calculate accurate orientation estimates.

Since the Hough transform only detects the existence of possible infinite lines in an image, we post-process the lines suggested by the Hough transform to find line segments.

To calculate the ego-motion we compare the classified lines that are found in two consecutive frames and if there is a match of one or more classified lines we then measure changes in gradiant and the distance from the robot to detected, classified lines. This yields a change in angle and distance from the robot, which is translated to robot movement and position information.

Figure 10) shows a acquired and processed image in the red arena. These images would correspond to frame1. Figure 10 a) Is the raw image b) is the edge detected image and c) is the result of the hough transform. Figure 10 d) shows the extracted line segments.

Figure 11 corresponds to frame2 after the robot has moved forward by a small amount. One can see that there is a change in the location of the detected lines in the image. This change is then translated into robot ego-motion and robot pose.

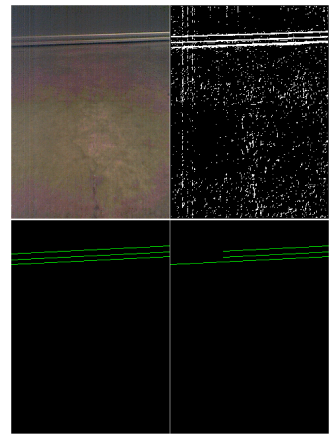

Figure 10: a) Raw image frame1 b) Edge detected image frame1 c) Lines extracted using hough transform d) Line segments extracted from hough space and edge detected image frame1

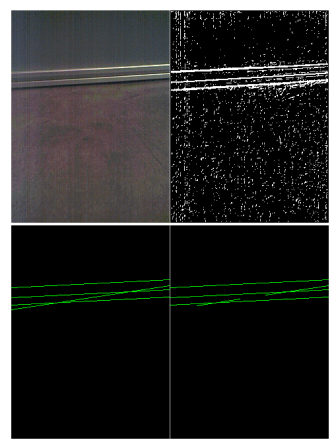

Figure 11: a) Raw image frame2 b) Edge detected image frame2 c) Lines extracted using hough transform d) Line segments extracted from hough space and edge detected image frame2

Ego Motion Estimation Using Optical Flow

The second part to our ego-motion measurement uses the optical flow of small patches in the view of the camera to determine translation and rotation from the previous to the current view. This approach is general in nature and does not rely on any specific features of the image itself.

In order to make ego-motion detection through visual feedback efficient for deployment on an embedded system, we begin by considering pixels in only a limited range of the image. We track small windows (8*8 to 16*16) in the image, and compute the optical flow within these windows to calculate the ego motion.

When analyzing the differences between images to determine optical flow, the approach begins by using the current model of the robot's motion to determine the most likely direction of the optical flow. The current motion of the robot is classified into four classes: straight forward, straight back, turn left, or turn right. Seven candidate patterns are generated that favor detection in the estimated direction of motion.

For example, consider the two images depicted in Figure 5. In this situation, commands recently sent to the robot were intended to cause it to drive in a straight line. The system therefore assumes that the scene depicted in the later (rightmost) image in the Figure should be shifted down relative to the scene depicted in the earlier (leftmost) image in the Figure. The system selects seven candidate patches to track which are arranged above the center of the image. Here, these regions have a good chance of appearing in the next image as patches close to the bottom of the image. Conversely, choosing patches on the bottom of the earlier image to track would not likely be helpful, as there is a strong likelihood that those patches would drop out of view. The candidate patches chosen for movement intended to be straight back, turning left, and turning right are similar to the ones shown in the example, but rotated by the obvious angle (90 or 180 degrees).

After selecting the patches to examine, the candidate patch with the largest cross-correlation difference is selected. The motivation is that this patch is most easily distinguished from the surrounding patches and thus the chance that it is tracked successfully is increased.

We use a Mini-max search method to find the patch with the largest cross-correlation difference. This is a useful and efficient technique. Once the first minimum has been established, the remaining candidate patches can be rejected even if only one of the cross correlations is larger than the current minimum. This is because the maximum of all cross correlations for a single patch is used. In the theoretical worst case, selecting the patch requires the computation of 30 cross-correlations. In practice, however, we found that this is rarely necessary and that the Mini-max selection finishes quickly. In the example, the red patch has the maximum minimum cross correlation to the other six patches and is therefore selected.

Having found the patch with the greatest cross-correlation difference, the system then finds a patch with a small cross correlation in the next image. This is done by estimating the optical flow and starting a search for the patch in the neighborhood of the predicted position of the patch in the new image.

The robustness of this method can be improved by tracking more than one patch and by computing a best estimate given the optical flow information for multiple patches. However, this would greatly increase the computational cost of the algorithm and its associated viability on small embedded systems. Therefore, we limited ourselves to tracking a single patch.

Stereo Vision

Our stereo vision system takes a new approach to determining how far away objects are to the robot.

The standard approach of stereo distance measurement is to identify feature *points* in the left and right image (Eg corners, start of line segments). This is problematic in the fact that these points are hard to identify and provide a weak match because they are highly susceptible to noise. Furthermore, the approach is computationally expensive since many possible point occurrences have to be considered.

The technique that we are currently working on is based on region identification. First regions are extracted separately from each image by growing regions based on detected edges, average colour and variance of neighbouring pixels in scanned lines. The scanned lines are then merged to create the individual regions. The regions are then filtered based on size (too small or too large) and shape (too thin) to yield candidate regions. Figure 12 shows the grown candidate regions of the left and right images. The regions are displayed by coloured boxes in each of the images, with different colours representing different regions. Regions in the separate images that have the same colour identify regions that have been found in both images simultaneously.

To calculate the robot's distance from objects we stereo match and calculate the disparity of regions based on similar shape and size, colour and variance. This yields a computationally cheap (fast) algorithm because there is usually only a couple of regions. It also provides robust regions matches, since regions are made up of a large number of individual pixels.

Conclusion

We have shown the overall system design of our robots that are involved in the USAR competitions (ZaurusBot and Spike) and have given a general overview of their system design.

We have also shown that ego-motion estimation for mobile robots using line tracking and optical-flow techniques has a great potential of reducing the error generated and increase the robustness of using classical sensing techniques or reduce the the number of sensors that are needed.

We have introduced a region-based stereo matching algorithm. The detected correspondences are more robust and the algorithm is computationally more efficient since it only considers matches between regions as opposed to feature points. In an average image, there are many more feature points than regions.

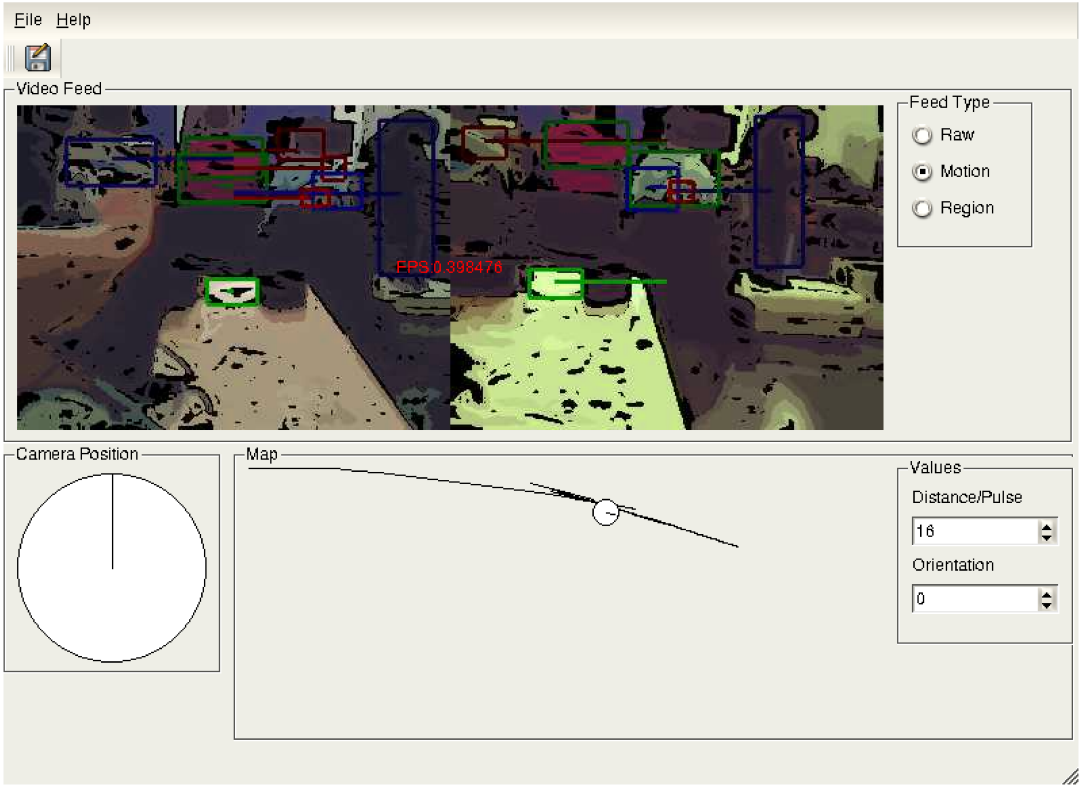

Figure 12: User interface displaying grown candidate regions in the left and right images

In the future we plan on combining the two ego-motion technique to finalize our ego-motion estimation toolkit and to also integrate the stereo depth information to develop a visual SLAM system that would localize a mobile robot in a structured or unstructured environment and produce an accurate three dimensional map.

References

Richard Szeliski, P. T. Geometrically constrained structure from motion: Points on planes.

Robert Sim, G. D. 1998. Mobile robot localization from learned landmarks. In *IEEE/RJS conference on Intelligent Robots and Systems*.

Stein, G. P.; Mano, O.; and Shashua, A. 2000. A robust vehicle ego-motion. In *Proceedings of IV-2000*.

Stephen Se, David Lowe, J. L. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks.

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