

# The Keystone Fire Brigade 2005

Jacky Baltes, John Anderson, Brian McKinnon, and Shawn Schaerer

Autonomous Agents Laboratory  
Department of Computer Science  
University of Manitoba  
Winnipeg, Manitoba, R3T 2N2, Canada  
jacky.andersj@cs.umanitoba.ca

**Abstract.** The Keystone Fire Brigade is a robotic rescue team that has previously competed in competitions at RoboCup (Fukuoka, '02; Padua '03, Lisbon '04), AAAI (Edmonton, '02; San Jose '04), and IJCAI (Acapulco, '03). The key elements of our approach are an emphasis on vision, a fully autonomous solution, and an implementation on inexpensive robot bases. This paper describes the version of the team that will be appearing at RoboCup-2005. We are using a fully autonomous robot with stereo vision for the first time. We overview the hardware employed, methods used for visual processing, map-making and victim identification. We also describe the experiences we had in the test domain and offer some recommendations on future competitions.

## 1 Introduction

Robotic rescue is both a worthwhile application for artificial intelligence and a challenge problem that allows solutions to be compared in a controlled setting. Robotic rescue pushed current research into autonomous robots to its limit. Most entries nowadays are teleoperated, because of the extreme difficulty of the USAR domain.

We believe that, ultimately, autonomous processing will be of great importance in robotic rescue. In rescue settings, issues such as operator fatigue, lack of situational awareness of the operator, cognitive load on the operator, and the number of individuals an operator can control in real time [1, 2] all place limitations on human control.

We also believe that a focus on autonomous processing is important from the standpoint of truly advancing artificial intelligence: the reason that most entries in the competitions are teleoperated is precisely because autonomous control is still very primitive in an area as complex as robotic rescue, and avoiding the use of autonomous control mechanisms does not do anything to improve this technology. We believe that once autonomy has improved, teleoperation can be added to fill in the gaps where a human operator can be helpful to the situation without being overwhelmed. We have thus been focussing on autonomy as a key focal point in our work in robotic rescue.

This paper describes the Keystone Fire Brigade the University of Manitoba's entry in the RoboCup-2005 Robotic Rescue Competition. Our approach embodies several principles we believe will ultimately be important in successful robotic rescue problems: autonomy, a multiagent perspective, and parsimony.

Parsimony is an important goal in all robotics applications. Any additional feature has a cost, both financially and in terms of computing power and other local resources,

and reliability. If a feature is not necessary to solve the problem, eliminating it provides more resources to those features that are necessary. We believe, like others [3] that the addition of any component should be carefully considered. Cost must be balanced with the efficacy of the equipment to the improvement of overall system performance. Parsimony is also one of the reasons that a multi-agent approach is important - by taking the same resources and spreading them among a number of simpler agents, the interaction of these over a geographic area can deal with the problem better than a single, highly complex agent. The more parsimonious the agent design, the more expendable any individual robot can be considered as well.

The remainder of this paper details the hardware platforms employed in this year's Keystone Fire Brigade the use of optical flow for localization and mapping, our region-based approach to stereo matching.

## **2 Team Members and Their Contributions**

We have a variety of people who have contributed to this year's team:

- Jacky Baltés: Team leader, lead designer, programmer
- John Anderson: Design, support
- Shawn Schaerer: optical flow
- Brian McKinnon: region segmentation, stereo matching
- Terry Liu: Programming

URL: <http://avocet.cs.umanitoba.ca>

## **3 Robot Hardware and Locomotion**

We have two primary motivations for hardware design in rescue robots. The first of these is reliance on extremely simple robotic platforms. Ultimately, the task of robotic rescue will benefit from implementation on inexpensive platforms, since larger teams can be afforded and individual robots can be viewed as being expendable in a dangerous environment.

Our motivation in using simple hardware, however, is to force reliance on more robust and versatile control methodologies. A system relying heavily on accurate odometry, for example, is severely crippled under conditions where odometry is inaccurate. A system that does not assume the availability of accurate odometry readings, however, will still operate under such conditions, as well as in conditions where odometry can be useful.

The second major design factor is an emphasis on vision. Each of our robots employs simple low-power CMOS cameras or webcams, and has enough local processing power for vision and robot control. Vision is the only sense employed by all of our robots.

In the 2005 version of the Keystone Fire Brigade two types of robots will be used. The first robot ZaurusBot is a simple differential drive design (see Fig. 1. Two servos attach to drive the left and right wheels, and an webcam is integrated for vision. The

servos were modified by cutting out the motion stop and thus provide relatively nice velocity control. Processing is provided by a PDA (Sharp Zaurus) with an attached CMOS camera.

The second robot is based on a toy car (see Fig: 1, and thus uses Ackerman steering. To support autonomous processing, the platform carries a VIA mini-ITX Eden 5000 microcontroller board with a 533 MHz Eden (x86 compatible) processor, 256 MB RAM, and a 256 MB flash card. We developed a mini Linux distribution (based on the Debian Linux distribution), which fits into this small space.

## 4 Sensors for Navigation and Localization

The greatest challenge on platforms such as those employed by the Keystone Fire Brigade is the design and implementation of pragmatic algorithms for intelligent visual processing, and the adaptation of these to the low frame rates that are achievable using the embedded systems driving the robots. This is the main contribution of our team.

The use of vision as the only form of sensing requires that vision not only be used to identify victims, which is the primary use of vision for most teams, but also to allow the robot to localize and map the environment. The following subsections describe our methods for dealing with each of these elements.

### 4.1 Ego Motion Estimation

In order for a robot using only vision to map an environment, its progress through the environment must be measured by vision rather than by odometry, sonar, laser, or a combination thereof. This is the problem of ego motion estimation, which requires the robot to estimate its movement in terms of distance and angle by examining the differences between visual frames.

we employ regular features for ego-motion estimation. Our approach uses the optical flow between images to estimate the motion of the robot (Figure 2), and also to indicate a lack of motion on the part of the robot (i.e. detecting when the robot is stuck).

If a recognizable pattern (a set of lines, intersections between lines, etc.) can be recognized in two different frames, we can compute the change in angle and distance on the part of the robot that produced that change in visual reference point. Note that we assume that the line is at a constant height (e.g., a line on the floor).

Figure 3 shows the geometry of the situation. Assuming that the robot can determine the angle between itself and a line, then the change in orientation  $\delta\theta$  can be easily computed by the difference in angle.

In the case of differential drive robot, this allows one to calculate the *difference* between the right and left wheel velocities (assuming the width of the robot is known). In the case of a rear-wheel or front wheel drive car, the steering angle can be computed (assuming the axle distance of the robot is known).

The change in angle of the line does not allow one to solve for right and left wheel velocities (in the case of a differential drive robot), or the linear velocity (in case of a car-like robot). However, given that the robot can also determine the distance between

**Fig. 1.** Robotics platform: ZaurusBot on the top, Spike on the bottom

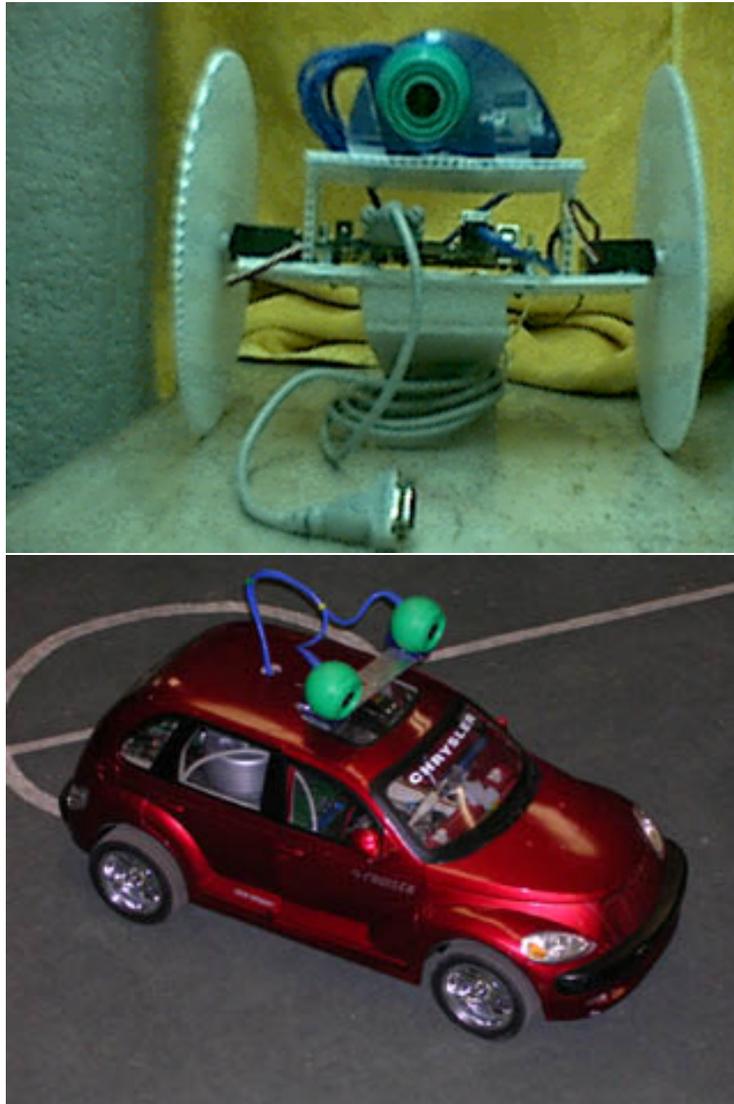
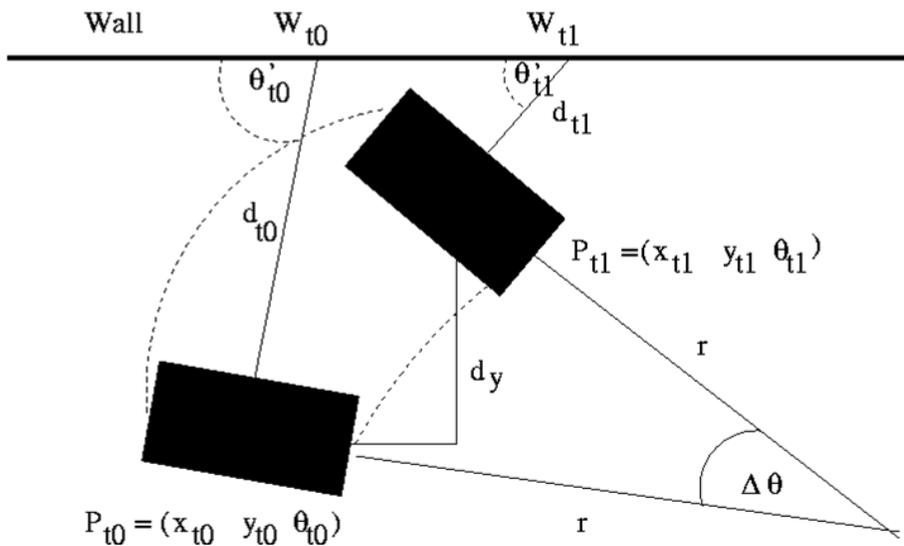


Fig. 2. Ego Motion Detection from Visual Frames



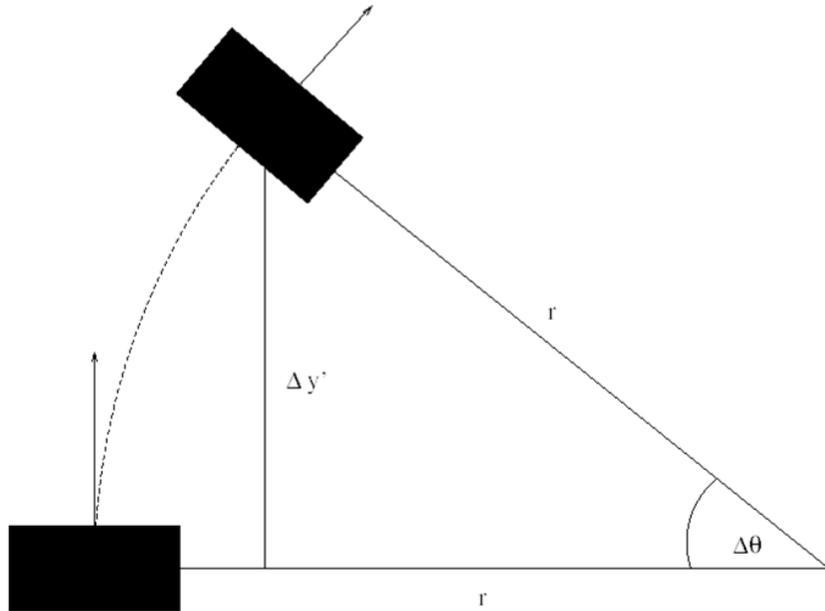
Fig. 3. Determining the change in angle from two visual reference points



the robot and the wall, solutions to the kinematic equations can be found and the motion can be recovered. The geometry and solution is shown in Fig. 4.

To determine if the robot is blocked or otherwise stuck in one position, the image is broken up into 16 equal sized sub-images. Of these, only the bottom 8 sub-images need to be considered - everything else is further away and would not be likely to provide useful feedback regarding the motion of the robot. The system then computes the differences between the current and the previous image for each quadrant. The colour difference is defined as the sum of the absolute value of the differences in the red, green, and blue channels. If the difference in a sub-image is above a threshold, the quadrant is marked. If there are more than eight marked sub-images and the motors were turned on in the previous time step, than the system signals that the robot is stuck or blocked. We break the image into sub-images to allow for localized changes due to the motion

Fig. 4. Determining the distance travelled from two visual reference points



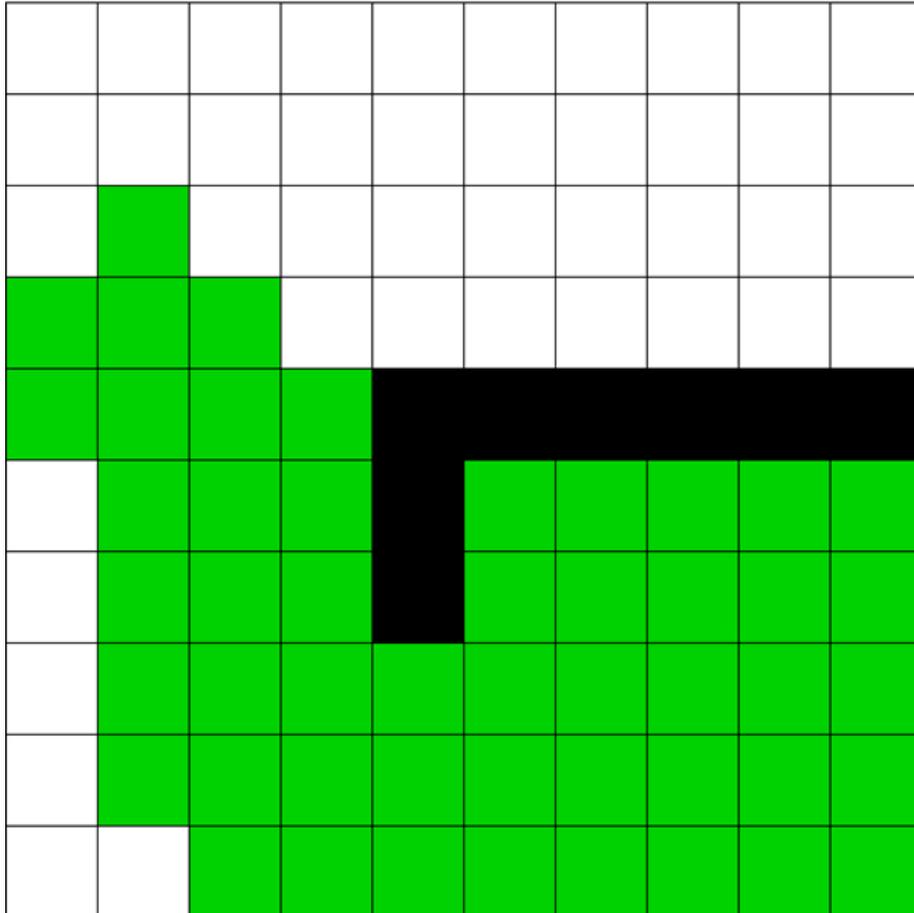
of some other agent or other external motion in the image, to try to limit the number of false positives.

## 5 Map Generation

Just as we rely solely on vision for localization through ego-motion detection, we also rely on vision for constructing a map while localizing through optical flow. This results in a chicken-and-egg problem: While localization becomes easier as maps are created, we must begin out of necessity with no valid initial map, making localization difficult, which in turn complicates the process of constructing an accurate map.

Our approach to building a map involves the construction of sets of local two-dimensional maps. The robot makes a map of the immediate area around itself (1m x 1m), storing the map as an occupancy grid such as that shown in Figure 5. In this map, the robot has plotted an obstacle (in black) and an open area (in green), while the white areas represent unexplored areas. These local maps are separated by longer traversals (a random walk in a particular direction) and are linked together as topological maps. The distance and length of a traversal serves as a link between maps, but as new features are detected earlier maps are studied for these features, allowing local maps to overlap. The smaller size of the local maps allows the local area to be explored

**Fig. 5.** Local Map



quickly, and the traversals between allow the robot to map different areas without errors in one map compounding to cause problems in later maps.

We plan in the future to extend this work to include a case-based reasoning system that employs typical sensor readings (especially “typical” images of the area) to identify areas and to connect them via topological paths.

## **6 Region Segmentation**

We have made significant progress in our work on stereo vision and in particular on using region segmentation to help stereo matching. The aim of our overall approach is to identify useful regions and match them between stereo images, with limited computa-

tional resources and under conditions typical of the USAR domain. In fully autonomous systems, stereo-matched regions are intended as input to routines for ego-motion detection, localization, and map-building (as employed originally in [4, 5] using a single camera).

We divide the process of performing region matching in stereo vision into six stages: color correction, image blur, edge detection, region extraction, region simplification, and stereo matching. Some of the more interesting stages will be described in the following subsections.

## 6.1 Edge Detection

After preprocessing the image with color correction and blurring, we use edge detection to find possible boundaries of regions.

We employ the Sobel edge detection in our implementation, because it is computationally simple and proves robust under a variable conditions. Sobel edge detection involves the application of convolution masks across the image. We employ two masks, for the horizontal and vertical dimensions respectively:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

After applying each mask, normalization is performed by dividing each pixel value by four. The resulting pixels are examined against a threshold value, where values larger than the threshold indicate an edge.

## 6.2 Region Growing

Having obtained a set of strong edges from a smoothed image, we use this along with the original smoothed image to determine a set of regions in each individual image.

Our approach to region segmentation involves growing regions from individual pixels using a stack-based approach. At any point, the pixels on the stack represent those from which future expansion of the region will take place. We begin by marking each pixel in the smoothed image as unexamined, and marking a single unexamined pixel as examined and placing it on the stack. We then repeatedly pop the topmost entry off the stack, and attempt to grow the region around it by examining the pixels immediately above and below, and to the left and right of that pixel. Each of these is tested to see if it is an edge pixel in the edge map, in which case it is ignored (allowing edges to form a strong boundary for regions). Each is also tested to see if it is a color match to the region being built, by summing the squares of the differences across all color channels. If this value falls below a defined threshold for error, the pixel is considered to be a part of the current region, and that pixel is placed on the stack to further extend the region; if not, the pixel is ignored. The threshold for color error is the mean color value of all pixels currently in the region, allowing the threshold to adapt as the region is grown. The algorithm terminates with a completed region once the stack is empty. A threshold

is set on the acceptable size of a grown region, and if the region size falls below this level, the region is discarded.

To extend this algorithm to grow a set of regions, we must recognize two things: first, it should be possible for an area to be part of more than one region in initial stages, since an image will generally be factorable into regions in a number of different ways. Thus, the algorithm must allow any pixel to be potentially claimed by more than one grown region. Second, once we throw a region away as being too small, we do not wish to start growing other regions within this same area, as this has already proved unfruitful. Similarly, once we have defined a region, it will be more useful to start new regions outside that defined area.

Our initial approach was to begin searching the image for a non-visited pixel, growing a region using the algorithm described above (while marking each pixel as examined when it is placed on the stack), and then starting the next region by searching for an unexamined pixel. This approach is functional, but in practice, linear scanning wastes resources because many unsuccessful regions are attempted. We have found it more fruitful to begin with randomly selected points (20 for a 320 x 240 image), selecting the location of each after regions have been grown from all previous points.

We also attempt to merge regions based on degree of pixel overlap. Each region is examined with others that it abuts or overlaps, and regions are merged if one of two thresholds are exceeded. The first of these is the percentage of pixels that overlap - this value requires a significant overall similarity, and is generally most useful in merging small regions. For merging larger regions, the likelihood of a large percentage overlap is small, and so the threshold used is a total pixel overlap. By using overlap rather than color separation as a basis for merging, shadows can be properly joined to the objects that cast them, for example, or glare to the objects the glare is placed upon, without having to set an excessively high color threshold.

At this point, we have a collection of strong regions in each of the two images (the top stereo pair in Figure 6). Each region is represented by a map between the original image and the region (a set of boolean pixels where each 1 indicates a pixel present in the image), as well as a set of region attributes: its size, mean colour value, centroid, and a bounding box.

### 6.3 Region Simplification

The next step in providing useful visual information to a robotic rescue agent is the matching of regions between a pair of stereo images. This, however, is a complex process that can easily consume a great deal of the limited computational resources available. Our initial stereo matching process involved examining all pixels that could possibly represent the same region across the stereo pair, requiring checking for a match between hundreds of pixels for each potential match. We have considerably simplified this process by simplifying the structure of the regions themselves, allowing us to match a much smaller set of data. This process is analogous to smoothing noise out of an image before looking for edges.

We simplify regions by generating a convex hull for each, allowing us to replace the set of points outlining the region with a simpler set describing a polygon  $P$ , where every point in the original point set is either on the boundary of  $P$  or inside it. We begin



**Fig. 6.** Segmented regions (top), with convex hulls plotted and distance lines from the centroid added (middle). Stereo-matched regions (bottom) are bounded by a colored box, with a line emanating from the centroid of the image

with the boolean grid depicting each image. The exterior points along the vertical edges (the start and end points of each row) are used to generate a convex hull approximating the region using Graham's Scan [?]. We form a representation for the convex hull by drawing radial lines at 5 degree intervals, with each line originating at the centroid of the region and extending to the hull boundary. The length of each such line is stored, allowing an array of 72 integers to describe each region. The middle stereo pair in Figure 6 illustrates the result of this simplification process.

#### 6.4 Stereo Matching

Once an image has been segmented into regions and simplified, regions must be matched across stereo images. Before simplifying regions, our original approach was limited in that it required superimposing region centroids and matching pixels. This was particu-

larly troublesome for large regions. With convex hull simplification, however, the efficiency of matching can be greatly improved. With each convex hull, the very first stored value represents the distance from the centroid to the hull boundary at the 0-degree mark. A comparison of the similarity of two regions can then be easily performed by summing the squares of the differences of the values in the 72 corresponding positions in the two arrays (implicitly superimposing the centroids). Beyond greatly decreasing the number of individual points to match, this representation allows time required to make a comparison independent of region size. There is no particular threshold to a match - each region is matched to its strongest partner in the corresponding stereo image. We do, however, constrain matches for the purposes of maintaining accuracy by forcing a match to be considered only after its appearance in three successive video frames. This is particularly useful for noisy and poorly lit environments such as USAR. The bottom stereo pair in Figure 6 illustrates the matching of three regions between the raw stereo sample pair. The lines in each region are used as an indication to a teleoperator the angle that one would region have to be oriented to match the orientation of the other. That is, straight horizontal lines require no reorientation. The use of these lines will be explained momentarily.

Since we are matching regions without regard to the location in the visual frame, similar regions can be matched despite unreasonable spatial displacement. This is equally true without employing convex hulls, and is part of the nature of this domain. Because the robot is moving over very uneven terrain, cameras are likely poorly calibrated, and as the domain is unpredictable, we cannot make strong assumptions about the position of a region in each of a pair of stereo images. If this were employed in a domain where such assumptions could be made, the process could be made more accurate by strongly constraining the distance between potential matches in regions in a stereo pair, thereby lowering the number of potential matches that would have to be considered.

## 6.5 Detection of Victims

While the system for awarding points is strongly oriented toward the use of multiple forms of sensing, most victims in the NIST testbed are reasonably easily identified visually. While there is much current work on the visual detection of victims, we are attempting to work with a reasonably simple, pragmatic approach to this difficult problem that is computationally viable for small embedded systems.

Our victim detection approach uses both colour as well as shape information. Flesh colored spots are marked as possible victim locations (these algorithms were trained beforehand on flesh patches of team members in the lighting used in the test arena).

We have developed a 12 parameter colour model which uses Red, Green, and Blue as well as three difference channels: Red - Green, Red - Blue, and Green - Blue. The differences are included in this parameter model because of their tendency to remain relatively constant in different views of objects of the same colour despite of lighting variations over a field. This approach is the same used in our Doraemon vision server [6] and has proven itself in years of robotic soccer competition.

Currently, we use a simple blob detection scheme. The system signals that it has found a victim by calculating the apparent size and aspect ratio of a skin coloured blob. If these parameters are within limits, the system signals the detection of the victim by

performing a series of 360 degree turns at the current location. It then continues to search the environment.

## **7 Practical Application to Real Disaster Sites**

While this base is extremely cheap and portable, this is not robust enough to employ in a real disaster setting. However, it is sufficient to demonstrate the approaches we believe are useful for future robotic rescue work, and can navigate within the yellow zone of the NIST testbed [7]. Our intent is to demonstrate the use of vision and autonomous control this task, and to further applied research in these areas in robotic rescue, rather than to tackle the less stable terrains in the orange and red arenas.

## **References**

1. Casper, J.: Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. Master's thesis, University of South Florida (2002)
2. Casper, J., Murphy, R.: Workflow study on human-robot interaction in usar. In: Proceedings of the IEEE International Conference on Robotics and Automation. Volume 2., Washington, DC, IEEE (2002) 1997–2003
3. Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. *Autonomous Robots* **1** (1994) 27–52
4. Anderson, J., Baltes, J., Kraut, J.: The keystone rescue team. In: Proceedings of the RoboCup Symposium, Padova, Italy (2003)
5. Baltes, J., Anderson, J.: A pragmatic approach to robot rescue: The keystone fire brigade. In: Proceedings of the AAI Workshop on Rescue Robots. (2002)
6. Anderson, J., Baltes, J.: The Doraemon User's Guide. Department of Computer Science, University of Manitoba, Winnipeg, Canada. (2002) <http://robocup-video.sourceforge.net>.
7. Jacoff, A., Messina, E., Evans, J.: Experiences in deploying test arenas for autonomous mobile robots. In Messina, E., Meystel, A., eds.: Proceedings of the 2nd Performance Measures for Intelligent Systems Workshop (PerMIS). NIST Special Publication 982, National Institute of Standards and Technology (2001) 87–95