X Workshop of Physical Agents (WAF'2009)

Pablo Bustos and Pilar Bachiller

***Abstract*—The Workshop of Physical Agents intends to be a forum for information and experience exchange in different areas regarding the concept of embodied agents, especially applied to the control and coordination of autonomous systems: robots, mobile robots, industrial processes or complex systems. This special issue is devoted to the selected papers presented at the WAF09 that took place from September 10th to 11th in the city of Cáceres (Spain).**

***Index Terms*—WAF, Physical Agents.**

I. The 10th edition of the Workshop of Physical Agents

THIS issue of the Journal of Physical Agents (JoPhA) is devoted to the selected papers presented at the X Workshop of Physical Agents (WAF'2009) that took place from September 10th to 11th in the city of Cáceres (Spain). Since its first celebration in 2000, the Workshop of Physical Agents has favored the meeting of different research groups from different areas of knowledge and with a common interest: physical agents. There are many groups who have taken part in this event and belong and actively participate in RedAF (the Spanish Network of Physical Agents), the main local sponsor organizing this Workshop. This annual event has three main objectives:

- Providing a forum for communication on research, technological innovation and technology transfer in the field of physical agents, encompassing areas such as embedded systems, artificial intelligence, mobile robotics, building automation and ubiquitous computing systems.
- Facilitating communication among researchers from different research groups related to physical agents.
- Providing a means for active and efficient transfer of research and technological upgrading.

In its tenth edition, the Workshop of Physical Agents has successfully achieved these goals once again. We had the participation of different research groups from the national scene who presented the results of their research in areas related to physical agents. We had contributions describing scientific works in different areas that use agent-based technologies concepts, robotics, vision, (software agents) highlighting the similarities and synergies among physical and software agents. This year we also had the participation of Professor Danica Kragic from the Royal Institute of Technology (Stockholm, SWEDEN) who gave the plenary talk *Active Vision for Detecting, Fixating, Manipulating Objects and Learning of Human Actions*. As in former editions of the workshop, there were shown live demos and exhibitions to assess progress in some research, such as soccer robots and autonomous navigation using vision.

Pablo Bustos and Pilar Bachiller are with University of Extremadura.
E-mail: {pbustos, pilarb}@unex.es

As a summary of this event, in this publication we include six papers about different topics of vision, SLAM, exploration and applications.

The first one *Multi-cue Visual Obstacle Detection for Mobile Robots* presents an algorithm for obstacle detection using a stereo camera pair. The method uses both geometric and color cues to perform robustly on heterogeneous circumstances. Authors apply the propose method to the problem of autonomous navigation showing results of real long-term experiments.

The second one *Recognition of Standard Platform RoboCup Goals* shows a methodology based on artificial vision for goal detection and its application to the RoboCup. The goals are detected by means of color segmentation and geometrical methods. Authors propose methods for estimating the robot position with respect to the goal solving problems as the self-localization of the robot in the field.

The third paper of this issue *Large Scale Egomotion and Error Analysis with Visual Features* presents a method for large scale robot 6DoF movement estimation using stereo images. The method transforms 2D features, based on Sift and Surf detectors, to the 3D space. These features are then used for robot movement correction. The paper shows results of a comparison between the proposed method and the ICP algorithm.

The fourth selected paper *Mission Specification in Underwater Robotics* describes the utilization of software design patterns and plan-based mission specifications in the definition of missions for autonomous underwater vehicles (AUV). Authors present the main elements of a software framework for programming AUVs, showing a solution to some well-known problems of the development of robotic control software.

The fifth one *Balanced Multi-Robot Exploration through a Global Optimization Strategy* proposes an original strategy for coordinating multi-robot exploration behaviors. The paper presents an extensive review of the state of the art in multi-robot exploration as well as a deep comparative analysis with other exploration techniques.

The last paper of this issue *Affine image region detection and description* proposes an affine region detector using a hierarchical grouping approach generating an irregular pyramid. In addition, detected regions are characterized by a kernel-based descriptor. Authors present a comparative study showing the performance of the proposal in relation to similar methods.

To conclude this introduction, we would like to thank all the authors for their excellent contributions and to extend our gratitude to all the participants of the X Workshop of Physical Agents.

Multi-cue Visual Obstacle Detection for Mobile Robots

Luis J. Manso, Pablo Bustos, Pilar Bachiller and José Moreno

***Abstract*—Autonomous navigation is one of the most essential capabilities of autonomous robots. In order to navigate autonomously, robots need to detect obstacles. While many approaches achieve good results tackling this problem with lidar sensor devices, vision based approaches are cheaper and richer solutions. This paper presents an algorithm for obstacle detection using a stereo camera pair that overcomes some of the limitations of the existing state of the art algorithms and performs better in many heterogeneous scenarios. We use both geometric and color based cues in order to improve its robustness. The contributions of the paper are improvements to the state of the art on single and multiple cue obstacle detection algorithms and a new heuristic method for merging its outputs.**

***Index Terms*—autonomous robots, visual navigation, obstacle detection.**

I. Introduction

OBSTACLE detection is one of the most fundamental needs for an autonomous navigation system to work. In order to avoid obstacles, the majority of approaches use laser or sonic range sensor devices. While sonic sensors are imprecise, short-sighted and usually unreliable, lidar devices are expensive. Nowadays, autonomous robots are usually provided with stereo camera pairs in order to perform tasks such as object recognition or visual SLAM. Thus, by using camera pairs also for obstacle detection, the cost of a laser device can be saved up. While vision systems can detect objects in their whole visual field, common laser sensors only sweep a plane, so those objects not intersecting the plane are missed. Two-axis sweeping lidars or integrated image and lidar sensors are even more expensive. Besides, lidar sensors are not error free, they may get wrong measures on shiny, or black objects not reflecting light as expected[8].

Experimental results show that neither appearance or geometric vision approaches are enough by themselves. Previous geometric ground-obstacle classifiers produce a high rate of false positives (e.g. in the borders of the paper sheet of figure 1) if they are not provided with a precise stereo calibration. Our stereo algorithm, as most geometric approaches, relies on the homography induced by a *locally* planar ground. The matrices coding the homography can be estimated in real time using the information from the stereo head configuration, or they can be manually set for a small set of known stereo head configurations. Our method is able to work within a considerable uncertainty range, performing well even with inaccurate calibrations. By relying on both geometric and appearance information in an appropriate way, our algorithm achieves higher reliability than those just working with one type of information. In addition, since our method does not use range sensor devices, it can detect floor downward discontinuities in addition to obstacles lying on the floor.

There has been previous research on visual detection of obstacles based on: color [5], [13] and geometric[2], [4], [12] information. In most of the color based approaches a hue histogram is created and used to classify pixels as obstacle or as free space. This is useful on very restricted environments, but leads to frequent false positives when approaching planar objects lying on the floor so that the robot can actually walk over them (see figure 1), and to false negatives as well (see figure 2). Color based approaches are not useful on grounds of heterogeneous colors. Geometric, homography based, approaches do not perform well with non-textured obstacles (see figure 3).

The most similar works we have found are [6], [15], where both color and geometric cues are used. However, they do not present any real improvement over single cue approaches. They suggest to combine both methods using *OR/AND* operators. If using an OR operator, it will fail where any of its cues get a false positive. If using an AND operator, it will fail where any of its cues get a false negative. Thus, despite its simplicity, this kind of cue integration is doomed to fail because it does not take into account the nature of its cues. In addition, [6] follows the same color-based classification as [5], which accuracy is discussed on section III.

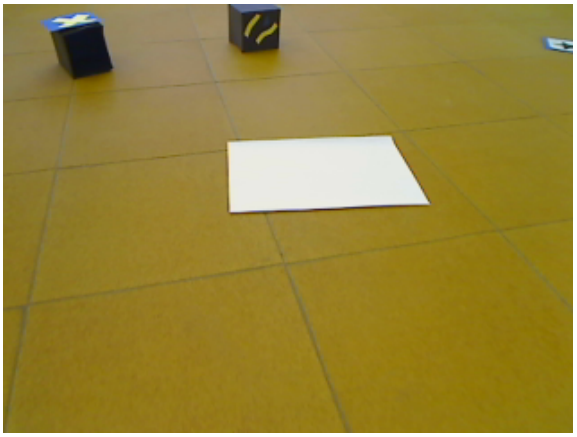


Fig. 1: Color-only approaches can lead to false positives.

Luis J. Manso is with University of Extremadura.
E-mail: lumafe04@alumnos.unex.es
Pablo Bustos is with University of Extremadura.
E-mail: pbustos@unex.es
Pilar Bachiller is with University of Extremadura.
E-mail: pilarb@unex.es
José Moreno is with University of Extremadura.
E-mail: josemore@unex.es

Fig. 2: Color-only approaches can lead to false negatives.

Fig. 3: Geometric-only approaches can lead to false negatives.

Our algorithm uses two obstacle detectors, based on two different visual cues: an appearance cue (color-based) and a geometric one (based on plane reprojection). Both of them produce a binary image in which white areas represent obstacles. The stereo cues are gathered in two steps: the first one is similar to [6], and the second one filters false positives from the output of the first step. The color-based algorithm also takes two stages: the first one classify pixels as obstacle if their color is not present enough in the histogram (in a similar way as seen in [5]), the second one filters false positives caused by noise. We also suggest a better alternative to the AND/OR operators for merging the binary output images. In addition, improvements to each of the two single cue obstacle detectors are presented.

The rest of this paper is organized as follows: Sections II and III describe the classification process using each of the single cue algorithms, geometric and color-based respectively. Section IV describes how the fusion of cues is made. Section V presents the robot platform used, the experiments performed and their results. Conclusions are detailed in section VI.

II. Geometric-based detection

The geometric-based obstacle detection algorithm is based on the idea that the floor is approximately planar. Given this assumption, the floor induces a planar homography between the two camera images. A planar homography is a projective geometry transformation that maps projected 3D points between two image planes assuming that they rest on a particular plane[1]. Pixels are mapped by premultiplying its homogeneous coordinates by the *homography matrix*:

$$x' = Hx, \tag{1}$$

so that for any pixel position in an image, a new position is determined for the other view point. In other words, the homography allows us to compute how a plane would be viewed from another perspective. When the intrinsic and extrinsic parameters of the cameras are known, the homography matrix can be calculated by using the following equation[1]:

$$H = K'(R - tn^T/d)K^{-1}, \tag{2}$$

where:

- K and K' are the intrinsic parameters matrices of the initial and the new viewpoints, respectively.
- R and t are the rotation matrix and translation vector that lead from the initial viewpoint to the new one, respectively.
- n is a normalized vector perpendicular to the plane inducing the homography.
- d is the distance from the initial viewpoint to the plane.

If the cameras of the robot are whether fixed or its only degree of freedom is a common pan movement, that movement will not change the homography, and explicit knowledge of the intrinsic or extrinsic parameters is not actually needed. In that case, it is possible to use different methods to directly estimate the homography matrix for a static configuration [2], [3], [7], [9], [10].

The initial ground-obstacle classification is done by warping one of the views to the other and comparing the result of the warping with the actual image seen in the last point of view. Under ideal conditions, provided that the camera can be modeled by the pin-hole model, every warped pixel corresponding to the floor would have the same value in both images. However, in real conditions, we have to face the following problems:

- light reflections.
- camera-camera desynchronization.
- camera-head position sensor desynchronization.
- different camera responses to the same color.
- not perfectly planar floors.
- stereo head pose uncertainty.
- imprecise homography estimation.

In [6], pixels are classified as obstacles if the difference between its values in the warped image and the actual one is above a threshold. The quality of this method relies heavily on the accuracy of the homography, a floor free of light reflections, and good lighting conditions. If the homography accuracy is not good enough, false positives often appear on edges.

Our method divides the classification in two stages in order to improve the reliability by including a second test. The first step of our method is similar to the previously seen ([6]), the

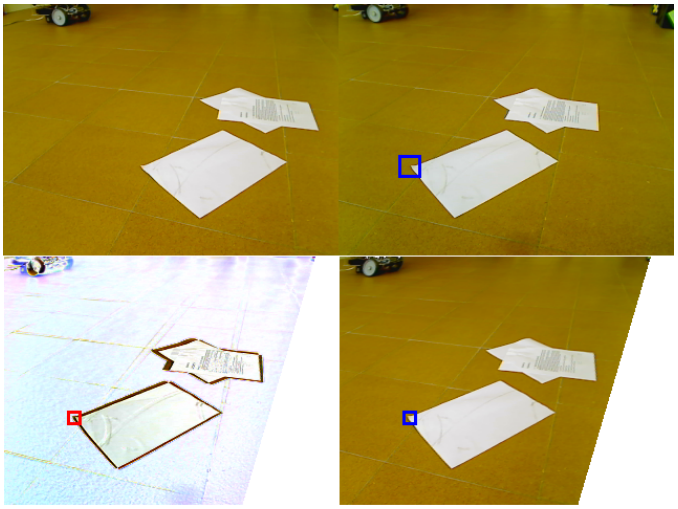


Fig. 4: Upper-left corner: left image. Upper-right corner: right image, destination point of view. Lower-right corner: left image homography-warped to the destination point of view. Lower-left corner: Absolute value of the right image minus the wapred image. Windows that do not pass the first test (as the one in red) are verified through a cross-correlation test (image patches in blue).

only difference is that we resize the input image for the first stage to reduce the computational cost. Thus, each output pixel represents a window in the actual input image. Those windows classified as obstacle by the first stage are then verified or discarded by the second one.

The aim of the second stage is to filter false positives. In order to do this, it is tried to find a match for the image patches of the destination point of view image corresponding to windows classified as obstacle by the first stage within a double-sized window in their correspondant position in the warped image. The match is only searched within a slightly bigger window in the same position because the image is already warped (under ideal conditions there would be a perfect match for every floor-window in exactly the same position). It is outlined in figure 4. This comparison is carried out by computing maximum output for cross-correlation. If the maximum for any of the channels is under a threshold, the window does not pass the test and it is classified as obstacle. Color information allows us to differentiate between different colors that have the same luminance. Here, a relatively low threshold should be used. The **key idea** is that it is safer and more stable to decide based on multiple low thresholds of different nature that complement each other, than using a single highly tuned one.

Since only candidate windows are tested, this second stage improves significantly the classification process without adding too much computational overload. This improvement not only decreases the occurrences of false positives on edges of textured floors, which is the biggest disadvantage of similar techniques, it also allows to dismiss low objects that do not actually represent an obstacle for the robot. In any case, no matter how approximate they are, there will be two different homographies, the actual homography that the floor induces in the two cameras, and the homography we are estimating. The second step allows these two homographies to be reasonably different without compromising the system.

Unlike shadows, which do not depend on the location of the observer, light reflections do. While shadows are just low illuminated areas, light reflections are a different phenomenon in which the floor acts as a mirror. Thus, light sources are imaged on different positions depending on the point of view of the camera, just like any object reflected in a mirror would. Unfortunately, this highly common phenomenon makes reflected light sources appear on a non-floor altitude. Even though they might not be detected and identified as proper reflections, and despite they can ruin the geometric classification we can still deal with them due to:

- robots do not actually walk over mirrors, reflections are diffuse
- the use of polarizing filters reduce reflections
- the second test reduces the impact of the remaining reflections

The main improvement of this algorithm to previous approaches is that it is partially immune to inaccurate homographies. Little variations on the actual homography produce mostly a translation of the image and little projective deformation. For every window passing the first test, the best correlation match is searched in its neighbourhood. If a window is not an actual obstacle it should have a good correlation match and should be discarded by the second test. As outlined before, this problem is very common when using mobile robotic heads. The scenarios in which this might be helpful are:

- **Loose or wrong camera position estimation:** Small translations or rotations of the robotic head, due to looseness or small impacts, may change the actual homography. In such cases, other approaches would give lots of false positives.
- **Motorized stereo configurations:** When the homography is recalculated in real-time in a stereo system, wrong pose and angle estimations, due to the imperfections of hardware or software, may lead to inaccurate homographies. This would also lead to false positives if using other approaches.
- **Camera desynchronization:** Even slight camera desynchronizations often make previous algorithms useless. This might or might not be common depending on the hardware used.

Once the second test is finished a binary image where pixels represent small windows of the original images is obtained. Assuming that the destination point of view is the one of the right camera, the process can be summarized as follows:

1) **Copy** the right image, I^R, to a temporal image I^{T1}.
2) Use H to **warp** the left image, I^L, to I^{T1}.
3) Compute the **absolute value** of I^{T1} - I^R. Store the result in I^{T2}.
4) For each window having a pixel over a threshold:
   a) **If** the maximum value for the normalized cross-correlation value in a bigger window is over a

threshold for all channels, set the window as floor.

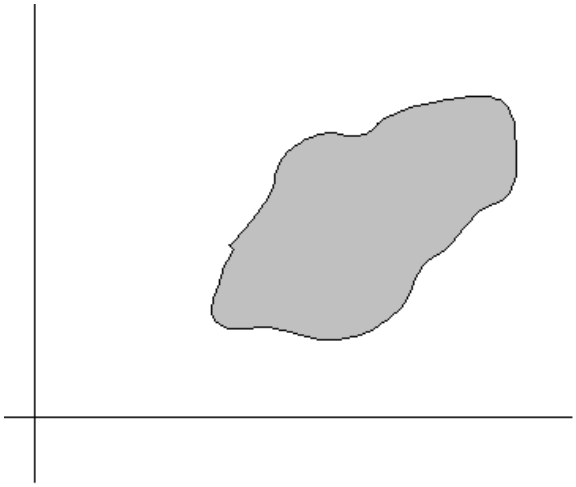
b) **Else** set the window as obstacle.

The first step can be skipped if, after the algorithm is finished, pixels outside binocular space are ignored. Optionally, if a window size resolution is not enough, a flood-fill operation can be started on the windows that passed the second test after resizing the output image.

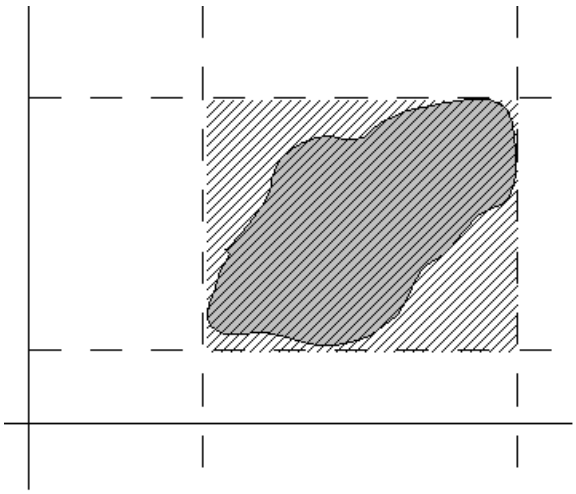
III. Color-based detection

The color-based obstacle detection is inspired on the approach seen in [5], pixels are classified based on their presence in a histogram which is obtained after a training process. The training consists in the selection of several image regions of the ground and the computation of a three dimensional histogram from those image regions. Region selection can be done manually by a human operator or, if the environment obstacles are textured (e.g. there are no untextured walls), by the robot itself selecting floor regions using the geometric-based classification previously detailed.

Instead of using two separate 1-dimensional histograms as suggested in [5], we use a single 3-dimensional one. The use of a three dimensional histogram is justified in terms of discrimination power. A 3-dimensional histogram is more powerful than a 1-dimensional one (i.e. a 3-dimensional histogram will always have, at least, the same information as a 1-dimensional one) and it is in no means harder to use. Even several threshold cuts on different 1-dimensional histograms will always sum up to axis paralell decision boundaries on corresponding multidimensional histograms. Despite the size of the histogram goes from n to n^3, by using 3-dimensional histograms we dramatically improve the color classification, without considerable time implications. Figures 5a and 5b show the difference of distriminative power graphically.



(a) Discriminative power of a 2-dimensional histogram.



(b) Discriminative power of two 1-dimensional histograms.

Fig. 5: Comparison between the discriminative power of a 2-dimensional histogram and two 1-dimensional histograms. This can be extended to a third dimension. The striped area of figure delimits the boundary of two 1-dimensional histogram while the grey area delimits the frontier of the 2-dimensional histogram.

We build our 3-dimensional histogram with values for normalized red and green components and an approximated value of luminance. Depending on the number of bins in which an axis is divided, we will get a different invariance for the values stored on it. The more bins an axis has the less invariance it gets. In particular, luminance invariance is desirable to some extent, but a complete invariance would entail a loss of discriminative power.

A reasonable way to build the histogram is to divide the X and Y axis in 128 bins and 32 bins on the Z axis. Thus, assuming that pixels are represented as RGB bytes (so their values range from 0 to 255), the X, Y and Z axis values would correspond to:

$$X: \quad 128R/(R+G+B+1) \tag{3a}$$

$$Y: \quad 128G/(R+G+B+1) \tag{3b}$$

$$Z: \quad (R+G+B)/12 \tag{3c}$$

Once the histogram is generated, it is low-filtered using a 3x3x3 mask and the training is finished.

Despite most color based obstacle detectors use HSV color space, we decided not to use it because hue values are usually very noisy at low saturation or low luminance. While other approaches as [5], [6] do not take into account pixels with low saturation, experimental results proved that greyish pixels should not be ignored. Obviously, the floor or some of the obstacles may be gray. Our three dimensional approach histogram shows better performance in the experiments.

In [15] a similar approach is taken, but it uses a two dimensional histogram, with normalized red and green values only. Thus, it is fully luminance invariant, which, as previously seen, is rarely a desired feature.

As well as in the geometric cue, the classification is divided in two stages. In the first stage, each pixel is classified according to the presence of its color in the histogram. If the quantity is lower than a threshold, the pixel is classified as obstacle. This threshold can be set as a percentage of the histogram population so it does not depend on the size of the training set.

In the second step of the color-based classification the output is windowized by taking only the pixel with the

lower value for each window. For this purpose 4-pixel width windows are used (i.e. a 4x4 window is classified as obstacle only if every pixel of the window have been classified as obstacles). This step removes false positive produced by noise.

IV. Multiple cue obstacle detection

An obstacle detection system based on only one of the described methods would perform well under certain circumstances: planar floor and textured obstacles in the geometric approach; and disjoint color sets for obstacles and floor in the color-based one. Since these conditions are seldom found, the quality of such classifications can be improved by using both at the same time, taking advantage of the different properties of the cues. A color-based only classification would often lead to non desirable classifications such as classifying as obstacle a paper sheet lying on the floor. On the other hand, a geometric-based only classification would not see any obstacle in an untextured wall. We suggest using both cues in order to produce a higher-quality classification. Nevertheless, we also claim that merging the results of both classifiers should be carried in a more sophisticated way than *AND/OR* operators.

Both, planar perspective mapping and color based cues are merged in order to get a single binary image as output, where each pixel is related to one of the windows of the single cue classifiers. As we will see in section V, the fusion of cues of different nature allows the robot to navigate through unstructured as well as structured environments. The only restriction is that the ground must be *locally* planar.

The cue fusion process is ruled by the two following principles:

1) Geometry wins on small obstacles:
   Here the main assumption is that a region classified as obstacle by its color, that is not classified by the geometry-based obstacle detector, is probably a planar object lying in the floor (or part of it) which does not actually represent an obstacle for the robot. Thus, unless the second principle tells the opposite, they should not be classified as obstacle. According to this, isolated regions classified as obstacles by their color, and not by the geometry-based obstacle detector, are ignored.
2) Color wins on large tall obstacles:
   In indoors environments walls are very common. Since usually they are untextured, the planar perspective mapping approach might not classify them correctly. Big regions reaching the **horizon line** are suspicious enough to assume they are untextured walls (or any other obstacle). This kind of regions are classified as obstacles despite they are not detected by the geometry-based algorithm. Thus, the whole connected area is classified as obstacle.

Following this two principles the visual field of the robot is divided in two areas, the one under the horizon line and the one above it. In the lower zone the geometric cue prevails, while in the upper one it is the opposite. In case that the connected-area of a region classified as obstacle by its color comprise part of both zones, the whole connected area is classified as obstacle. This particular aspect can be seen in figure 6.

It is reasonable to think that there is no point classifying by their color the image zone above the horizon line because it is already known that its pixels correspond to obtacles assuming a planar floor. Nevertheless, because connected-areas may span part of both zones, the method provides new information. This is specially important in order to avoid walls or dodge tall untextured obstacles.

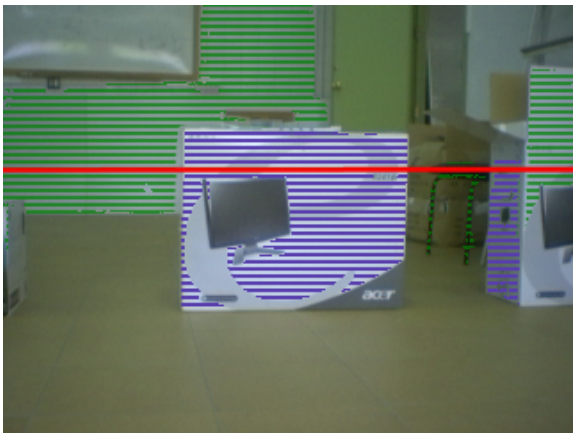


Fig. 6: This picture shows the horizon line (in red) and the connected-areas classified by their color that span part of the under-horizon area (dashed).

The horizon line of a plane can be calculated as the image line going through two different vanishing points of lines belonging to the same plane. According to [1], the vanishing point of a line is the intersection of a parallel line passing through the camera center with the image plane. In [1] it is demonstrated that under a projective camera $P = [K|I]$ the vanishing point v of a line is imaged as:

$$v = Kd \tag{4}$$

being d a bound vector representing the direction of the line. As can be deduced, for a particular camera, the vanishing poing of a line only depends on its direction. Thus, in order to get two different vanishing points of lines belonging a particular plane, it is necessary to use intersecting lines.

Given a vector n' normal to the plane of interest π (usually $(0,1,0)$) and a camera K whose extrinsic parameters are known, such directions can be easily calculated. The extrinsics parameters are needed to transform n' to the frame of reference of the camera, hereafter n. Thus, the two directions can be computed as:

$$d1(x) = +2000 \tag{5a}$$
$$d2(x) = -2000 \tag{5b}$$
$$d1(z) = 10000 \tag{5c}$$
$$d2(z) = 10000 \tag{5d}$$
$$d1(y) = d1(z)n(z)/n(y) + d1(x)n(x)/n(y) \tag{5e}$$
$$d2(y) = d2(z)n(z)/n(y) + d2(x)n(x)/n(y) \tag{5f}$$

V. Experimental results

The described system leads to a binary image which can be easily used for navigation. The floor boundary can be calculated by scanning the columns of the binary image bottom-up,

constructing a polyline with the first occurrences of obstacle pixels in the image. Also, if the extrinsic camera parameters are known, assuming that obstacle pixels are near the floor, the binary image can be mapped into world coordinates. With the binary image mapped to world coordinates, a laser measure can be estimated and thus, local navigation can be solved by using any of the existing range-based algorithms such as VFH+ or Potential Fields[11].

Figures 8, 9 and 10 are examples of the output of the obstacle detection system. The left obstacle of figure 8 and both obstacles of figure 10 are detected by both geometric-based and appearance-based algorithms. The wall at the right of figure 8, only has texture at its bottom, which actually lies on the floor plane, thus, it can only be detected by its appearance. The system is also able to discard the planar objects lying on the floor which appear on figures 9 and 10. In all of these figures floor plane is represented by light colored lines and obstacles are represented by dark ones.

The rest of the section will introduce the robot platform used and the results of the experiments which lead us to the conclusions that follow in section VI.

A. Description of the used robot platform

The obstacle detection system has been tested on RobEx, a low-cost differential robot platform developed at the Robotics and Artificial Vision Laboratory of the University of Extremadura, previously used and described in[16]. The cameras are two USB webcams working at 20Hz. Both cameras are mounted on a three degrees of freedom head system. The computation is done on-board in real-time by a laptop carried by the robot. Even using a low cost robot and cheap cameras, the robot is able to navigate autonomously in real-time on a wide variety of environments. The robot is shown in figure 7.

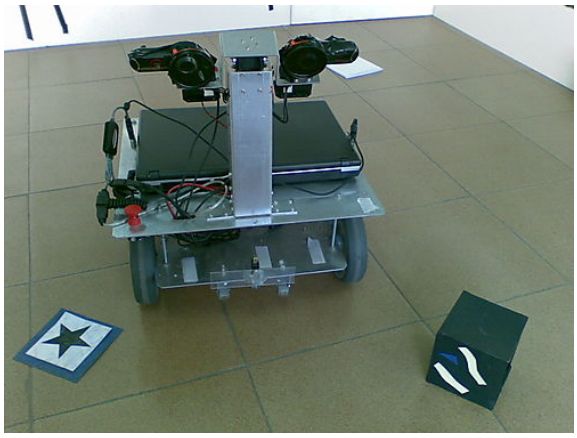


Fig. 7: RobEx is the robot we used in our experiments.

As seen in section II, light reflections depend on the point of view of the camera, so they may lead to a wrong classification. Despite the impact of light reflections on the geometry algorithm is minimized using the second test, it may not be enough depending on the material of the floor. A useful property of electromagnetic waves, such as light, is that when they are reflected in a planar surface, they get polarized at some extent. We use this phenomenon to decrease the appearance of these reflections on the images by providing the cameras with circular polarizing filters, which remove polarized light (e.g. light reflections on the floor). Reflections are closer to full polarization as they approach the so called *Brewster's angle*[14], which depends, in this case, on the refractive index of air and floor. Although light at angles far from the Brewster's angle are only partially polarized and hence the polarizing filters do not remove the reflections completely, they have been proved to be helpful.

B. Single vs. Multiple cue navigation

Single cue navigation was performed on both structured and unstructured environments. The results obtained with the color based approach outperformed the ones obtained with previous similar methods, but it is still not reliable by itself. Ground surfaces and obstacles sometimes share the same color, specifically when the scene is highly or barely illuminated.

Results obtained with the geometric approach clearly outperformed the ones obtained with previous similar methods. The overhead of the second test is negligible, as it is done only on candidate windows. In spite that the geometric obstacle detection is enough on unstructured environment where untextured image areas are very rare (with exception of the sky which is not an obstacle), on structured environments may be problematic.

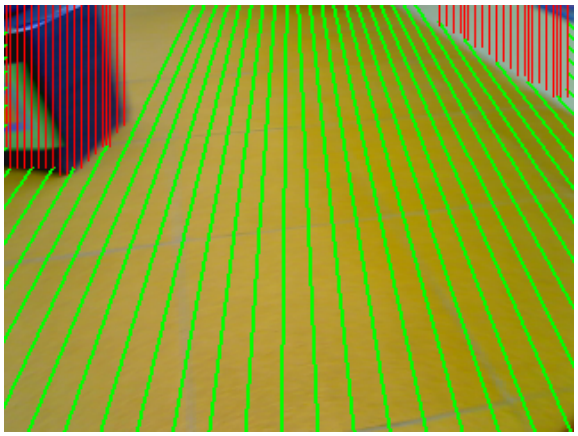


Fig. 8: The obstacle and the wall are detected.

C. Sixty minutes challenge using multiple cues

When the obstacle detection system development was started, a thirty minutes challenge was established as a measure of reliability. The challenge was passed in exceed when using cue fusion. In fact, it has passed several times a sixty minutes challenge. With geometric single-cue navigation, the challenge was also passed in unstructured environments (textured) but it failed in structured scenarios with untextured walls. On single cue navigation, the results depended heavily on the color of the ground and the obstacles.

The scenario in which the sixty minutes tests have been succesfully accomplished is a specially tailored environment

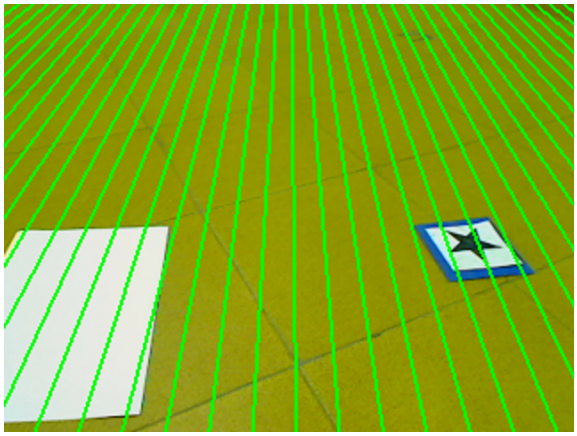


Fig. 9: Both, the paper sheet and the landmark, are discarded as obstacles.

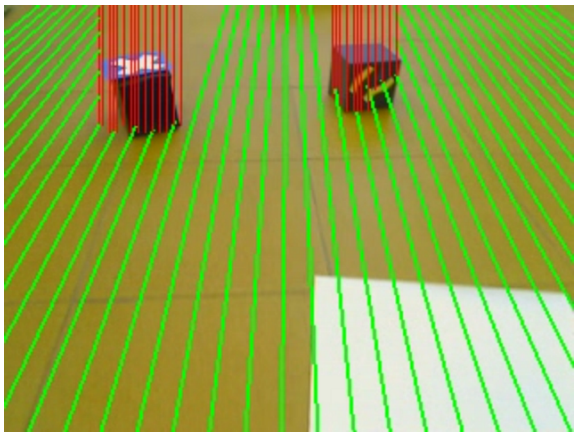


Fig. 10: Obstacles are detected, the paper sheet lying on the floor is discarded.

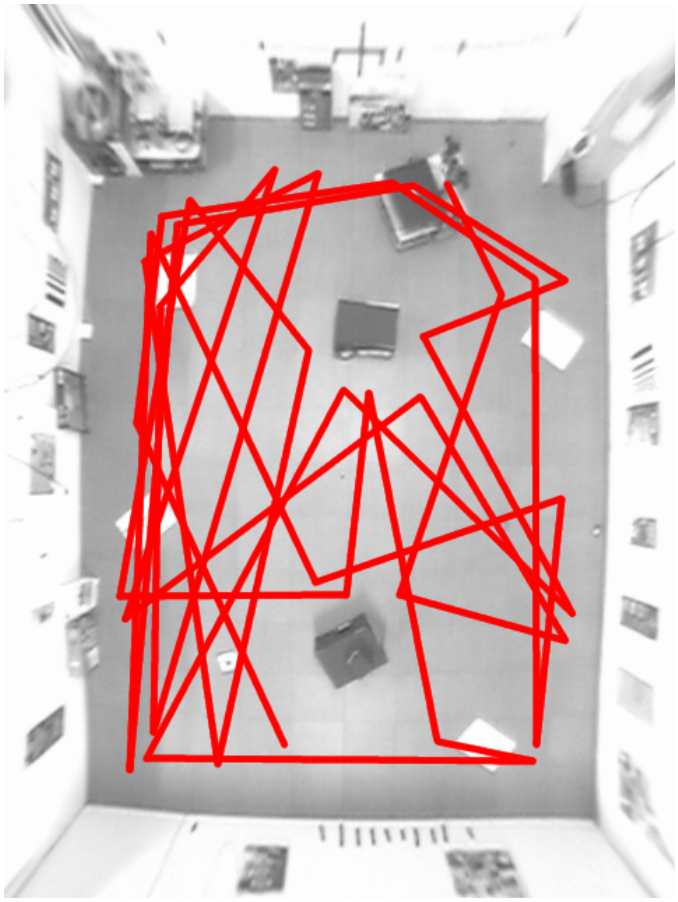


Fig. 11: RobEx robot trajectory using a naive wandering algorithm.

with tricky obstacles and non-obstacle objects with different heights, shapes and colors, as well as light reflections. A picture of this environment can be seen in figure 11, where the trajectory of a 7-minute wander is also shown.

D. Real time homography estimation

Due to the problems seen in II, specifically camera-camera desynchronization, camera-head position sensor desynchronization, and stereo head pose uncertainty, the approaches in [6], [15] are error prone when using motorized stereo heads. Because of the second test of the geometric classifier, our method discards false positives with reasonable desynchronizations or pose estimation errors.

In our experiments, the real time homography estimation performed successfully even with rough camera calibrations.

VI. Conclusions

In this paper we have presented two improvements to previous single cue detectors and a new technique for merging their outputs. The new single cue detectors, appearance and geometry based, outperform their previous counterparts. In addition, the proposed method for cue fusion clearly outperforms any single cue detector, as well as the multiple-cue method detailed in [6]. Due to the variety of obstacles present in the tests, we have been able to demonstrate that our approach allows the robot to navigate through a wide repertoire of conditions. The computation is light enough to be carried in real-time on an average laptop, coexisting with other heavier software components.

In order to improve significantly the system performance, our view is that it will be required a richer environment representation. The only situations in which the algorithm does not work as it would be desirable is when facing untextured obstacles with the same color as the ground (see picture 2) and with low height obstacles, no matter their color, with very little or no texture and a rounded shape (so that the obstacle is imaged in the same way as a planar object would). Generally these situations can be easily overcome with a dense, symbolic, geometric representation of the environment that can only be achieved using knowledge-based modelling techniques in conjunction with state-estimation algorithms. Steps are being taken towards this goal.

Acknowledgment

This work has been supported by grant PRI09A037 from the Ministry of Economy, Trade and Innovation of the Extremaduran Government.

REFERENCES

[1] Richard Hartley and Andrew Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2004.

[2] J. Gaspar and J. Santos-Victor and J. Sentieiro, "Ground Plane Obstacle Detection with a Stereo Vision System", in *International Workshop on Intelligent Robotic Systems*, 1994.

[3] Jorge Lobo and Jorge Dias, "Ground Plane Detection using Visual and Inertial Data Fusion", in *International Conference on Electronics Circuits and Systems*, IEEE, pp. 479-482, 1998.

[4] Zhou and Baoxin Li, "Robust Ground Plane Detection with Normalized Homography in Monocular Sequences from a Robot Platform", in *International Conference on Image Processing*, IEEE, pp. 3017-3020, 2006.

[5] Iwan Ulrich and Illah Nourbakhsh, "Appearance-Based Obstacle Detection with Monocular Color Vision", in *"in Proceedings of the AAAI National Conference on Artificial Intelligence*, AAAI Press, pp. 886-871, 2000.

[6] Parag Batavia and Sanjiv Singh, "Obstacle Detection Using Adaptive Color Segmentation and Color Stereo Homography", in *in International Conference on Robotics and Automation*, IEEE, pp 705-710, 2001.

[7] Y.I. Abdel-Aziz and H.M. Karara, "Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry", in *Proceedings of the Symposium on Close-Range Photogrammetry*, American Society of Photogrammetry, pp. 1-18, 1971.

[8] Wolfgang Boehler and Andreas Marbs, "Investigating Laser Scanner Accuracy", Leica Geosystems HDS, 2003.

[9] Jin Zhou and Baoxin Li, "Homography-Based Ground Detection for a Mobile Robot Platform Using a Single Camera", in *in International Conference on Robotics and Automation*, IEEE, pp. 4100-4105, 2006.

[10] Thomas Bergener, Carsten Bruckhoff and Christian Igel, "Evolutionary Parameter Optimization for Visual Obstacle Detection", in *in The International Institute for Advanced Studies in Systems Research and Cybernetics*, pp. 104-109, 2000.

[11] Iwan Ulrich and Johann Borenstein, "VFH*: Local Obstacle Avoidance with Look-Ahead Verification", in *in International Conference on Robotics and Automation*, IEEE, pp. 2505-2511, 2000.

[12] Darius Burschka, Stephen Lee and Gregory Hager, "Stereo-based obstacle avoidance in indoor environments with active sensor re-calibration", in *International Conference on Rob otics and Automation*, IEEE, pp. 2066-2072, 2003.

[13] Scott Lenser and Manuela Veloso, "Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision", in *International Conference on Intelligent Robots and Systems*, IEEE, pp. 886-891, 2003.

[14] A. Lakhtakia, "General Schema for the Brewster Conditions", in *Optik*, Vol 90, pp. 184-186, 1992.

[15] Yong Chao and Zhu Changan, "Obstacle Detection using Adaptive Color Segmentation and Planar Projection Stereopsis for Mobile Robots", in *International Conference on Robotics, Intelligent Systems and Signal Processing*, IEEE, pp. 1097-1101, 2003.

[16] Pilar Bachiller, Pablo Bustos and Luis J. Manso, "Attentional Selection for Action in Mobile Robots", in *Advances in Robotics, Automation and Control*, I-Tech, pp. 111-136, 2008.

Recognition of Standard Platform RoboCup Goals

José M. Cañas, Eduardo Perdices, Tomás González and Domenec Puig

***Abstract*—RoboCup provides a common research framework where a wide range of problems closely related to robotics and artificial intelligence must be addressed. The main focus of the RoboCup activities is competitive soccer. Thus, the visual identification of players and play-field components is a necessary task to be performed. In particular, goals are some of the key elements that should be identified by each player (robot). In this way, this paper introduces a fast and robust methodology based on Artificial Vision techniques for the recognition of the goals utilized in the RoboCup Standard Platform League. First, 2D images captured by the front camera mounted in the head of a Nao robot are used to recognize the goals through a color based geometrical segmentation method. Afterwards, the position of the robot with respect to the goal is calculated exploiting 3D geometric properties. The proposed system is validated with real images corresponding to the RoboCup2009 competition.**

***Index Terms*—RoboCup and soccer robots, Artificial Vision and Robotics, Nao humanoid, Goal Detection, Color Segmentation, 3D geometry.**

I. Introduction

ROBOCUP [1] is an international robotics competition that aims to develop autonomous soccer robots with the intention of promoting research in the fields of Robotics and Artificial Intelligence. It offers soccer as a dynamic, competitive and cooperative benchmark for testing the robotics technology and pushing it forward. Its long term goal is to build a soccer team of robots able to beat the human world champion team by mid-21st century. Maybe this is the equivalent to the long tearm milestone of AI community with artificial chess players, and Deep Blue defeated Gary Kasparov in 1997. The current state of the robotics technology is far from such ambitious goal, but progress has been made since the first RoboCup was celebrated in 1997.

In the last years several public challenges and competitions have arisen around robotics. For instance DARPA Grand Challenge and Urban Challenge have contributed to foster the research in robotics, providing proofs of concept about the feasibility of autonomous robot on real transportation missions.

RoboCup has worldwide scope and in the last years has included new categories beyond soccer: Junior, Rescue and Home. The last ones trying to reduce the gap between the contest and real applications. Several new leagues have also appeared around the soccer category, depending on the robot size and shape: small size, middle size, humanoid and standard platform league (SPL). Maybe the most appealing one is the SPL as the hardware is exactly the same for all participants. The behavior quality and performance differences lie completely in the software. In addition, the code of the robots must be publicly described, so the knowledge sharing pushes the overall quality. Until 2007 the hardware platform was the Sony Aibo. Since 2008 the SPL hardware platform is the Aldebaran Nao humanoid (Fig.1). Its main sensors are two non-stereo cameras and with it the teams have been exposed to the complexity of biped movement.

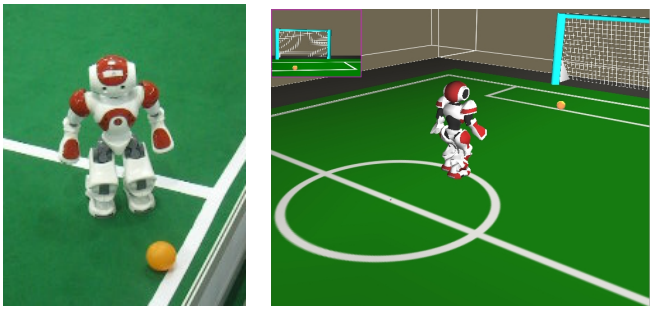


Figure 1. Nao humanoid and Webots simulator

In SPL, the robot players must be completely autonomous. In order to build a soccer robot player many different habilities must be programmed, both perceptive and motion or control oriented. For instance the goto ball behavior, the follow-ball behavior, the ball detection, the kicking, self-localization, standing up in case of fall, etc.

This work is focused in the goal detection, based on the camera images of the Nao. The goal detection helps the robot to decide whether to kick the ball towards the opponent's goal or just turn to clear the ball out of its own goal. It can also provide good information to self-localization inside the game field.

The rest of this paper is organized as follows. Section II reviews the state of the art in artificial vision systems in the RoboCup. In section III several solutions to the same problem of goal detection in the images are proposed. Section IV proposes a technique for obtaining spatial information from the previously detected goals. Section V shows experiments with proposed techniques. Finally, conclusions and further improvements in given in section VI.

II. Vision based systems in the RoboCup

Over the last years, considerable effort has been devoted to the development of Artificial Vision systems for the RoboCup soccer leagues. In this way, the increasing competitiveness and

José M. Cañas and Eduardo Perdices are with the Rey Juan Carlos University. E-mail: jmplaza@gsyc.urjc.es

Domenec Puig and Tomás González are with the Rovira i Virgili University. E-mail: domenec.puig@urv.cat

This work has been partially funded by projects RoboCity2030 (ref.S-0505/DPI/0176) of the Comunidad de Madrid, and by the Spanish Ministries of Education and Science under projects DPI2007-66556-C03-03 and DPI2007-66556-C03-01.

[1] www.robocup.org

evolution of the RoboCup leagues has conducted to vision systems with high performance, which are addressing a variety of typical problems [12], such as perception of natural landmarks without geometrical and color restrictions, obstacle avoidance, pose independent detection and recognition of teammates and opponents, among others.

Several constraints in the RoboCup domain make difficult the development of such vision systems. First, the robots always have limitated processing power. For instance, in the Nao humanoid a single AMD Geode 500Mhz CPU performs all the onboard computations and the Naoqi middleware consumes most of that capacity. Second, the robot cameras use to have poor quality. In the Aibos the camera was of 416x320 pixels and the colors were not optimal. Third, the camera is constantly in motion, not stable in height as the robot moves through the field.

A. Background

A number of initiatives for developing vision systems conceived to give solutions to the aforementioned typical problems have been carried out in recent years. In this line, early work by Bandlow et al. [8] developed a fast and robust color image segmentation method yielding significant regions in the context of the RoboCup. The edges among adjacent regions are used to localize objects like the ball or other robots on the play field. Besides, Jamzad et al. [10] presented several novel initiatives on robot vision using the idea of searching on a few jump points in a perspective view of robot. Thus, they performed a fast method for reliable object shape estimation without the necessity of previously segmenting the images.

On the other hand, the work by Hoffmann et al. [11] introduced an obstacle avoidance system that is able to detect unknown obstacles and reliably avoid them while advancing toward a target on the play field of known color. A radial model is constructed from the detected obstacles giving the robot a representation of its surroundings that integrates both current and recent vision information.

Further vision systems include visual detection of robots. In this sense, Kaufmann et al. [13] proposed a methodology that consists of two steps: first, the detection of possible robot areas in an image is conducted and, then, a robot recognition task is performed with two combined multi-layer perceptrons. Moreover, an interesting method presented by Loncomilla and Ruiz-del-Solar in [12] describes an object recognition system applied to robot detection, based on the wide-baseline matching between a pattern image and a test image where the object is searched. The wide-baseline matching is implemented using local interest points and invariant descriptors.

Furthermore, recent work proposed by Volioti and Lagoudakis [14] presented a uniform approach for recognizing the key objects in the RoboCup This method proceeds by identifying large colored areas through a finite state machine, clustering of colored areas through histograms, formation of a bounding boxes indicating possible presence of objects, and customized filtering for removing unlikely classifications.

B. Related work on visual goal detection

In general, the aforementioned research has been oriented to solve visual tasks in the environment of the RoboCup. However, some of those works have specifically proposed solutions to the problem of goal detection.

In this regard, one of the earliest approaches was given by Cassinis and Rizzi [9] that performed a color segmentation method using a region-growing algorithm. The goal posts are then detected selecting the boundary pixels between the goal color and the white field walls. After that, image geometry is used to distinguish between the left and the right goal post.

The aforementioned work described in [8] has been also applied to the detection of the goals in the RoboCup leagues. In this way, they detect goals by the size of the regions obtained after applying the color based image segmentation mentioned above. Moreover, [14] aims at recognizing the vertical goal posts and the goal crossbar separately. Both horizontal and vertical goal indications and confidence levels are derived from the horizontal and vertical scanning of the images, according to the amount of lines detected. Afterwards, it is decided whether the previously obtained indications can be combined to offer a single goal indication and, finally, different filters are used to reject unlikely goal indications.

III. Goal detection in 2D

Two different approaches oriented to the detection of the goals that appear in the 2D images are described in the next two subsections. The first one puts the emphasis on the geometric relations that must be found between the different parts that compose a goal, while the second is focused on edge detection strategies and specifically in the recognition of pixels belonging to the four vertices of a goal: Pix1, Pix2, Pix3 and Pix4 as shown in Fig.2.

A. Detection Based on Geometrical Relations

The first proposed method is intended to be robust and fast in order to overcome some of the usual drawbacks of the vision systems in the RoboCup, such as the excessive dependency of the illumination and the play field conditions, the difficulty in the detection of the goal posts depending on geometrical aspects (rotations, scale,...) of the images captured by the robots, or the excessive computational cost of robust solutions based on classical Artificial Vision techniques. The proposed approach can be decomposed into different stages that are described in the next subsections.

1) Color calibration: The first stage of the proposed method consists of a color calibration process. Thus, a set of YUV images acquired from the front camera of the Nao robot is segmented into regions representing one color class each.

Fig.2 shows an example image captured by the Nao robot containing a blue goal.

The segmentation process is performed by using a k-means clustering algorithm, but considering all the available centroids as initial seeds. Thus, in fact, seven centroids are utilized, corresponding to the colors of the ball (orange), goals (yellow and blue), field (green), robots (red and blue) and lines (white).

Figure 2. Example of original image from a RoboCup competition

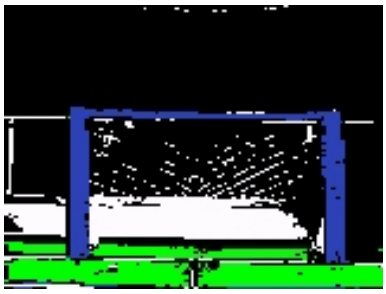


Figure 3. Color segmentation

The range between the minimum and the maximum YUV values in the regions obtained after that clustering stage are considered as the actual prototype values that characterize each color class of interest. Fig.3 depicts the color image segmentation produced by applying the range of color values automatically obtained through the calibration to the example image in Fig.2. The good segmentation results in Fig.3 indicate that the prototype values for each color of interest have been correctly determined during the calibration process.

2) Geometral and Horizon Planes Detection: The next step consists of the estimation of the geometral and horizon planes according to the robot head position. In order to do this, firstly, the pitch and yaw angles that indicate the relative position of the robot head with respect to the play field are calculated. On the one hand, the geometral plane is defined as the horizontal projection plane where the observer is located. On the other hand, the horizon plane is parallel to the geometral plane and indicates the level above which there is no useful information.

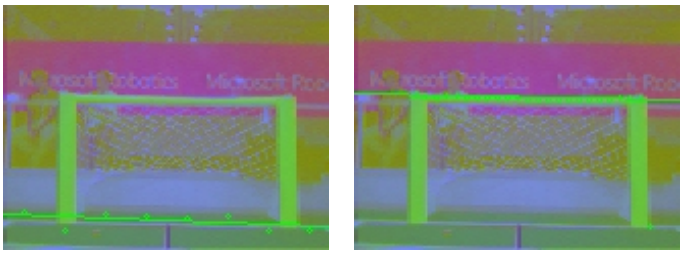


Figure 4. Intersection of geometral and horizon plane with image plane

Thus, the position matrix of the robot head is used for determining the horizontal inclination of the image with respect to the play field. Then, a grid composed of series of parallel vertical lines perpendicular to the horizontal inclination previously mentioned is calculated. The intersection between the grid and the green play field produces a set of points. The line across these points is the intersection line between the geometral plane and the image plane. In fact, the goal posts will be searched above this line. Fig.4 (left) displays the intersection between the geometral plane and the image plane corresponding to the example image in Fig.2.

Furthermore, intersections among the grid and the top blue or yellow pixels in the image are detected (taking into account the inclination of the image). The line across those points constitutes the intersection among the horizon plane and the image plane. It is expected not to find useful information in the images above this line. Fig.4 (write) displays the intersection between the horizon plane and the image plane in the example image in Fig.2. Note that, by definition, the geometral and the horizon planes are parallel and delimit the region where the goals are expected to be found.

3) Goal Posts Detection: The overall aim of this process is to extract the goal posts and other interesting features that could reinforce the detection of goals in the play field.

First of all, the color prototypes obtained as explained in Section III.A1 are used to segment the blue and yellow goal posts and crossbars. In order to do this, not all the image pixels are analyzed, but a high resolution sampling grid is utilized in order to detect blue or yellow lines in the image. Fig.5 depicts the detected lines corresponding to a blue goal (long lines correspond to the posts and short blue lines to the crossbar) corresponding to the example image in Fig.2.

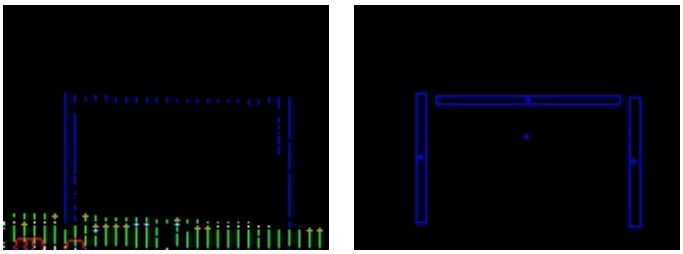


Figure 5. Interest points and goal blobs

In addition, a process to detect interest points is performed. The same grid mentioned before is utilized to detect crossings between blue or yellow lines (belonging to the goal posts) and white lines in the play field (goal lines). Also, crossings among green pixels (belonging to the play field) and white lines that delimit the play field are identified. If those interest points are detected close to the blue or yellow lines, previously sampled, they reinforce the belief that those lines belong to the goal posts. Red circles in Fig.5 (left) enclose interest points identified in the original image shown in Fig.2.

4) Goal Recognition: Once a set of pixels distributed into parallel lines corresponding to the goal posts and crossbar have been identified according to the procedure described in the previous section, the last step consists of a recognition process that finally locates the gravity center of the goal.

In order to perform such task, the aforementioned lines are grouped into blobs. A blob is composed of neighbor lines with similar aspect ratio (an example is shown in Fig.5 (right)). Finally, the blobs identified in this way are grouped into a perceptual unit that can be considered as a pre-attentive goal. Then, we apply an intelligent case reasoning strategy to bind

that unit into a coherent goal. Fig.5 (right) illustrates the blobs that configure the goal that appears in Fig.2 after it has been recognized by the proposed technique. The geometric center of the goal is also indicated according to the recognition method.

B. Detection based on color, edges and Hough transformation

We have also developed a second simple method to detect goals in 2D images. It follows four steps in pipeline. First, a color filter in HSV color space selects goal pixels and maybe some outliers. Second, an edge filter obtains the goal contour pixels. Third, a Hough transformation gets the goal segments. And fourth, some proximity conditions are checked on the vertices of such segments, finding the goal vertices Pix1, Pix2, Pix3 and Pix4. All the steps can be shown at Fig.6.

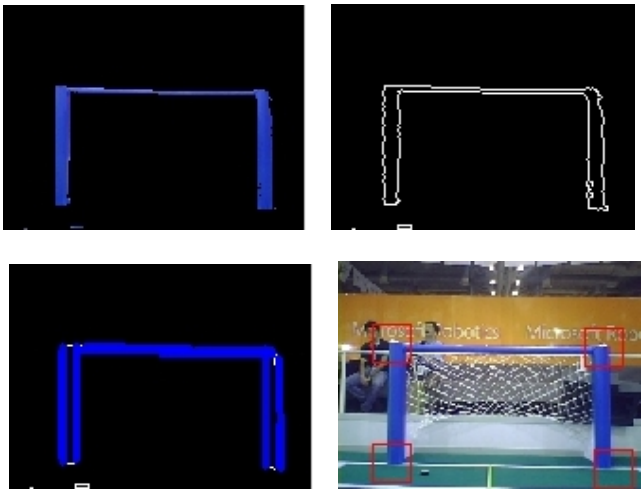


Figure 6. Goal detection based on color, edges and Hough transformation

IV. Goal detection in 3D

Once the goal has been properly detected in the image, spatial information can be obtained from that goal using geometric 3D computations. Let Pix1, Pix2, Pix3 and Pix4 be the pixels of the goal vertices in the image, which are calculated with the algorithms of section III. The position and orientation of the goal relative to the camera can be inferred, that is, the 3D points P1, P2, P3 and P4 corresponding to the goal vertices. Because the absolute positions of both goals are known (AP1,AP2,AP3,AP4) that information can be reversed to compute the camera position relative to the goal, and so, the absolute location of the camera (and the robot) in the field.

In order to perform such 3D geometric computation the robot camera must be calibrated. Its intrinsic parameters are required to deal with the projective transformation the camera does over objects in 3D world when it obtains the image. The pinhole camera model has been used, with the focal distance, optical center and skew as its main parameters. In addition, two different 3D coordinates are used: the absolute field based reference system and the system tied to the robot itself, to its camera.

We have developed two different algorithms to estimate the 3D location of the perceived goal in the image. They exploit different geometric properties and use different image primitives: line segments and points.

A. Line segments and thorus

Our first algorithm works with line segments. This algorithm works in the absolute reference system and finds the absolute camera position computing some restrictions coming from the pixels where the goal appears in the image.

There are three line segments in the goal detected in the image: two goalposts and the crossbar. Taking into consideration only one of the posts (for instance GP1 at Fig.2) the way in which it appears in the image imposes some restrictions to the camera location. As we will explain later, a 3D thorus contains all the camera locations from which that goalpost is seen with that length in pixels (Fig.8). It also includes the two corresponding goalpost vertices. A new 3D thorus is computed considering the second goalpost (for instance GP2 at Fig.2), and a third one considering the crossbar. The real camera location belongs to the three thorus, so it can be computed as the intersection of them.

Nevertheless the analytical solution to the intersection of three 3D thorus is not simple. A numerical algorithm could be used. Instead of that, we assume that the height of the camera above the floor is known. The thorus coming from the crossbar is not needed anymore and it is replaced by a horizontal plane, at h meters above the ground. Then, the intersection between three thorus becomes the intersection between two parallel thorus and a plane. The thorus coming from the left goalpost becomes a circle in that horizontal plane, centered at the goalpost intersection with the plane. The thorus coming from the right goalpost also becomes a circle. The intersection of both circles gives the camera location. Usually, due to simmetry, two different solutions are valid. Only the position inside the field is selected.

To compute the thorus coming from one post, we take its two vertices in the image. Using projective geometry and the intrisinc parameters of the camera, a 3D projection ray can be computed that traverses the focus of the camera and the top vertex pixel. The same can be computed for the bottom vertex. The angle α between these two rays in 3D is calculated using the dot product.

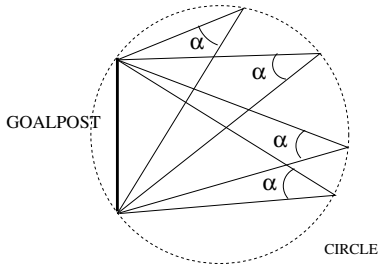


Figure 7. Circle containing plausible camera positions

Let's now consider one post at its absolute coordinates and a vertical plane that contains it. Inside that plane only the points in a given circle see the post segment with an angle α. The

thorus is generated rotating such circle around the axis of the goalpost. Such thorus contains all the camera 3D locations from which that post is seen with a angle α, regardless its orientation. In other words, all the camera positions from which that post is seen with such pixel length.

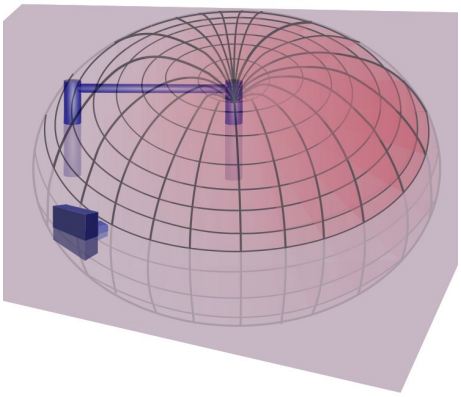


Figure 8. Thorus containing all plausible camera positions

B. Points and projection rays

The second algorithm works in the reference system tied to the camera. It uses three goal vertex pixels Pix1, Pix2 and Pix3. For Pix1, using the pinhole camera model, a projection ray R1 can be drawn which traverses the camera focus and contains all the 3D points which project into such Pix1. R2 and R3 rays are computed in a similar way, as seen in Fig.9. The problem is to locate the P1, P2 and P3 points into their corresponding projection rays.

Assuming that we know the position of P1 in R1 then only a reduced set of points in R2 and R3 are compatible with the real goal size. Because the distance between P1 and P2 is known (D12), P2 must be in R2 and the sphere centered at P1 with D12 radius, named S2 (Fig.9). The general intersection between R2 and S2 yields two candidate points: P2' and P2" (there can also be no interesection at all or only one single point). Following the same development and the distance D13 between P1 and P3, two more candidate points are computed: P3' and P3".

Combining those points we have several candidate tuples $(P1, P2', P3')$, $(P1, P2'', P3')$, $(P1, P2', P3'')$ and $(P1, P2'', P3'')$ All of them contain points located at the projection rays and all of them hold the right distance between P1 and the rest of points, but the distance between P2 and P3 may not be correct. Only the real solution provides good distances between all of its points. A cost function can be associated to choose the best solution tuple. We used the error in distance between P2 and P3, compared to the good distance D23.

In fact, the P1 position in R1 is not known, so a search is performed for all the possible P1 values. The algorithm starts placing P1 at λ distance from the camera location. All the candidate solution tuples are calculated and their costs computed. For each λ the cost of its best tuple is stored. The search algorithm explores R1 increasing λ at regular intervals

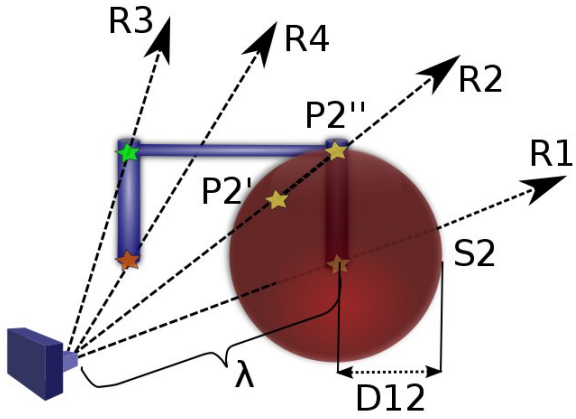


Figure 9. Projection rays for the four goal corners

of 10cm, starting close to the camera location and up to the field size, as can be seen at Fig.10. The tuple with the minimum cost is chosen as the right P1, P2 and P3 values. P4 is directly computed from them. They are the relative 3D position of the goal in the camera reference system.

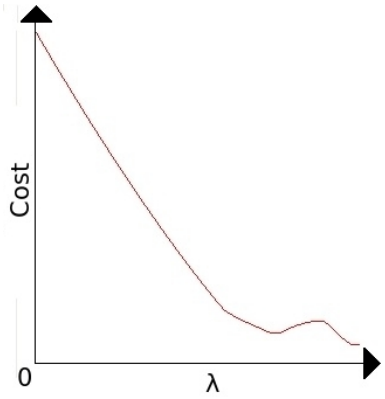


Figure 10. Cost function for different λ values

Finally, the absolute 3D camera position can be computed from $(P1, P2, P3, P4)$. Because the absolute positions of the goal in the field reference system are known $(AP1, AP2, AP3, AP4)$, we can find a rotation and translation matrix RT that fits the transformation of P1 into AP1, P2 into AP2, etc. We have used the algorithm in [1] for that. The estimated translation represents the absolute position of the camera in the fied based reference system.

V. Experiments

Several experiments have been carried out to validate our algorithms, both in simulation and with real images. For simulated images we have used Webots (Fig.1) and for real ones a benchmark of images collected from the Nao's camera at the RoboCup2008 and RoboCup2009, placing the robot at different field locations.

The first set of results presented in this section correspond to the 2D goal detection strategy presented in Section III-A. In particular, Fig.11, Fig.12, Fig.13 and Fig.14 display

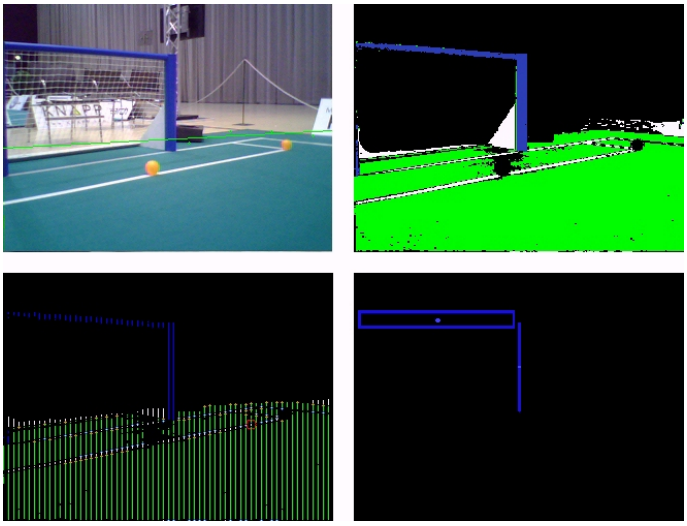


Figure 11. Experiment1: Goal detected with method described at Section III-A

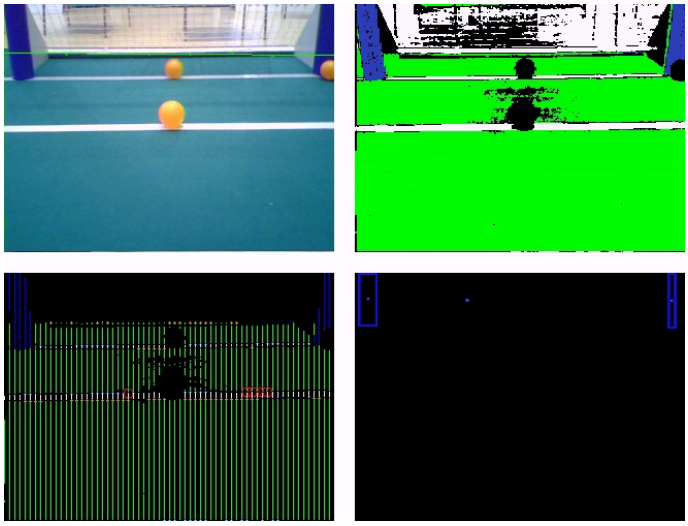


Figure 13. Experiment3: Goal detected with method described at Section III-A

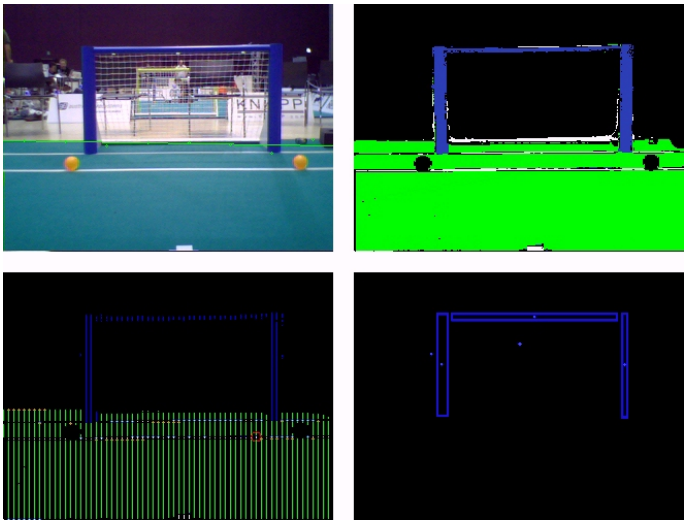


Figure 12. Experiment2: Goal detected with method described at Section III-A

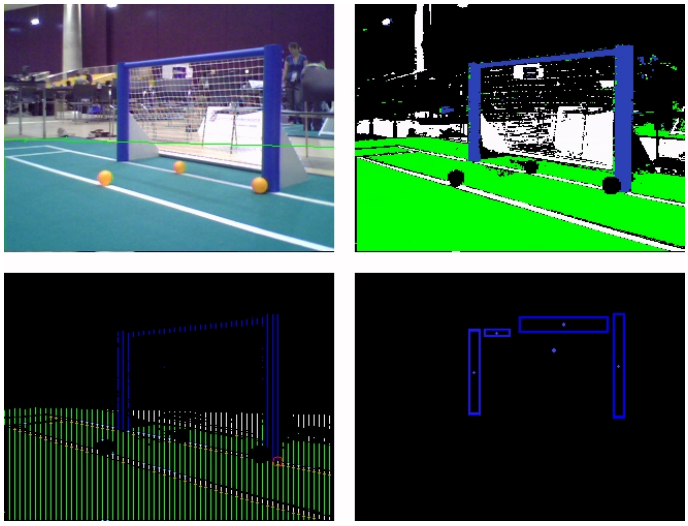


Figure 14. Experiment4: Goal detected with method described at Section III-A

four examples corresponding to real RoboCup images and the results produced by the different steps of the proposed method. These experiments have been run on a Pentium IV at 3.2 GHz and the average computation time is 9ms. The left images in the first row of each figure depict the original color image including a green line that represents the intersection between the geometral plane and the image plane. The right images in the first row show the color segmentation. Every left image in the second row of each figure displays a grid composed of series of parallel lines perpendicular to the horizontal inclination of the image (red circles enclose interest points). Finally, right images in the second row depict the recognized goal and its gravity center for each example image. As it can be appreciated, the proposed strategy is able to recognize goals even in situations involving certain difficulties, such as when only a small part of a goal appears in the image, or if the play field is not visible in the image, or when the goal is seen from a side in the play field.

A second set of results produced by the technique explained in section IV are shown in Fig.15. The input images are displayed at the right side. The output of the 2D detection is overlapped as red squares at the goal vertices (Pix1,Pix2,Pix3,Pix4). In the left side of the figure the field lines and goals are drawn, and the estimated camera position is also displayed as a 3D arrow.

In order to measure the accuracy of the goal detection algorithm in 3D, the robot has been placed at 15 different field locations and the estimated relative distances in XY plane between the goal and the robot have been compared to the real ones, as shown in the table I. In this table all the positions and errors are in centimeters, and the goal is centered at (0,200). The mean error is below 13cm for the thorus based method (presented at section IV-A) and below 20cm for the projective ray method (described in section IV-B). This error includes the effect of the non ideal calibration of the Nao camera, both in its intrinsic parameters and its height. Another interesting

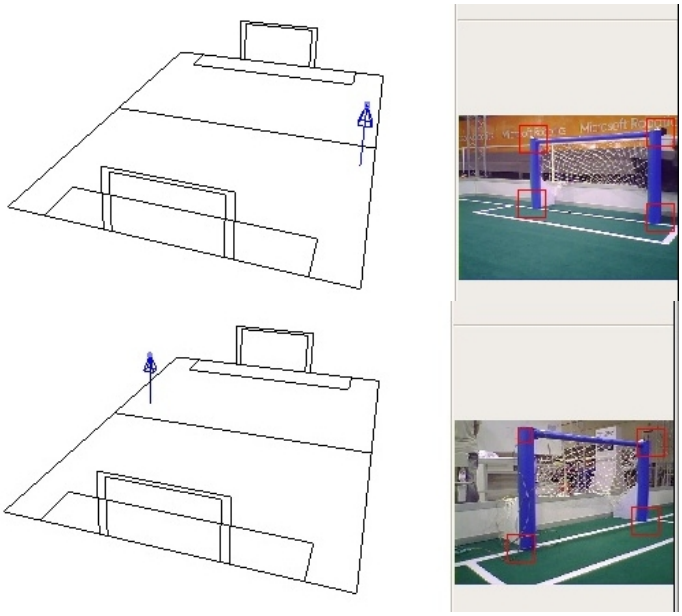


Figure 15. 3D position from the goal detected in the image

Point (X,Y)	Error with thorus	Error with projective rays
centimeters		
P1 (100,400)	9.4	10.5
P2 (250,400)	10.3	11.6
P3 (450,400)	20.1	15.8
P4 (450,200)	19.5	38.7
P5 (250,200)	13.2	27.5
P6 (180,200)	6.2	19.7
P7 (100,0)	9.1	9.6
P8 (250,0)	10.8	17.8
P9 (450,0)	19.4	33.5
P10 (180,350)	1.2	15.2
P11 (250,350)	11.3	9.2
P12 (180,50)	8.4	15.7
P13 (250,50)	15.4	18.3
P14 (300,140)	4.8	26.7
P15 (300,260)	11.5	21.3
Mean Error	13.1	19.4
Typ. deviation	5.5	8.8

Table I
ACCURACY

result is that error increases as the distance to the goal grows, as expected, but a good estimation is achieved even from the furthest side of the field as can be seen in Fig.16 (P3, P4 and P9 points).

Processing step	PC	Nao
2D detection based on Hough transformation	13'2ms	250ms
Thorus based 3D detection	1ms	1ms
Projective rays 3D detection	2ms	33ms

Table II
TIME CONSUMPTION

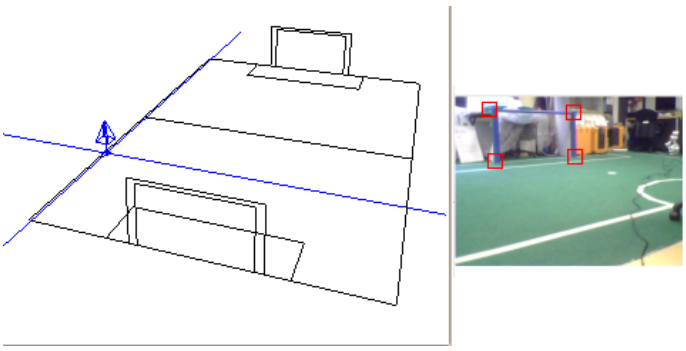


Figure 16. 3D goal detection from a far point (P9)

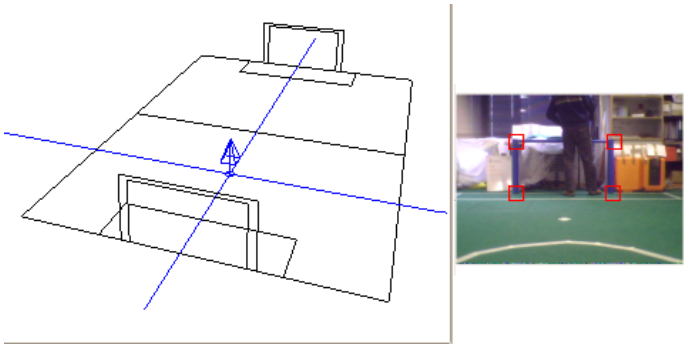


Figure 17. 3D goal detection with a partial occlusion of the goal

The experiments presented in this paper have been obtained both processing the images in the onboard Nao's computer and processing the real images offline in a 3 GHz Pentium-IV machine. The time consumption corresponding to both 2D and 3D proposed techniques are shown in table II, where the algorithms have been evaluated in both the Pentium (PC column) and the Nao's computer (Nao column). In particular, times for each of the processing steps to detect the goal in 3D are shown. As it can be seen, the 3D algorithm is the fastest. The projection rays method is slower than the thorus method, maybe because it is a search algorithm. For edge filter and Hough transformation we have used OpenCV library.

The algorithm performs well both with the ideal images coming from the Webots simulator and the real images from the Nao at RoboCup-2008 and RoboCup-2009. In the case of the 2D goal detection at section III-B the color filter must be properly tuned for each scenario. The 3D techniques are robust to partial occlusions of the goal in the image as long as the four corners are properly detected, as shown in Fig. 17.

VI. CONCLUSION

Vision is the most important sensor of the autonomous robots competing at the RoboCup. Goal detection is one of the main perceptive habilities required for such autonomous soccer player. For instance, if the humanoid had the opponent's goal just ahead it should kick the ball towards it. If the goal in front of the robot was its own goal, then it should turn or clear the ball away. In this paper two different algorithms have been proposed to detect the goal in the images coming from the robot's camera. The first one is based on geometral and horizon planes. The second one uses an HSV color filter, an edge filter and Hough transformation to detect the post and crossbar lines.

In addition, two new methods have been described which estimate the 3D position of the camera and the goal from the goal perceived inside the image. The first one uses the line length of the posts and intersects two thorus to compute the absolute 3D camera position in the field. The second one uses the projection lines from the vertice pixels and searches in the space of possible 3D locations. They locate the goal in 3D with an error below 13 and 20 cm respectively. Both are fast enough to be used on line inside the humanoid's computer. This 3D perception is useful to the self-localization of the robot into the field.

All the algorithms have been implemented as a proof of concept. Experiments have been carried out that validate them and the results seem promising as shown in Section V.

We are working on performing more experiments onboard the robot, with its limited computer. We intend to optimize the implementation to reach even better real time performance, in order to free more computing power to other robot algorithms like navigation, ball perception, etc. required for proper autonomous operation.

The proposed 3D algorithms assume the complete goal appears in the image, but this is not the general case. The second future line is to expand the 3D geometry algebra to use the field lines and incompletely perceived goals as source of information. For instance the corners and field lines convey useful self-localization information too.

REFERENCES

[1] A. Lorusso, D.W.Eggert and R.B.Fisher, *A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations*, Proceedings of the 1995 British conference on Machine vision, Vol. 1, pp 237-246, Itech Education and Publishing 1995.

[2] Thomas Rofer et al., *B-Human, Team Report and code release 2008*

[3] S. Lenser and M. Veloso, *Visual sonar: fast obstacle avoidance using monocular vision* in Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2003. (IROS 2003), pp 886-891, 2003.

[4] M. Jüngel, *A Vision System for RoboCup*, diploma Thesis, Institut für Informatik, Humboldt-Universität zu Berlin, 2004.

[5] J.C. Zagal, J. Ruiz-del-Solar, P. Guerrero and R. Palma, *Evolving Visual Object Recognition for Legged Robots* in RoboCup 2003: Robot Soccer World Cup VII, LNCS 3020, pp 181-191, 2004.

[6] D. Herrero and H. Martínez, *Embedded Behavioral Control of Four-legged Robots* in Robotic Soccer, Pedro Lima editor, pp 203-227, 2007.

[7] RoboCup Standard Platform League (Nao) Rule Book.

[8] T. Bandlow et al., *Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario* in M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI, vol. 1856, pp. 174–185, 2000.

[9] R. Cassinis and A. Rizzi, *Design Issues for a Robocup Goalkeeper*, in M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI, vol. 1856, pp. 254–262, 2000.

[10] M. Jamzad, E.C. Esfahani, S.B. Sadjad, *Object Detection in Changing Environment of Middle Size RoboCup and Some Applications*, Proc. IEEE Int. Symp. On Intelligent Control, Vancouver, Canada, pp. 807-810, 2002.

[11] J. Hoffmann, M. Jungel, and M. Lotzsch, *A Vision Based System for Goal-Directed Obstacle Avoidance used in the RC'03 Obstacle Avoidance Challenge*, 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences), LNAI, vol. 3276, pp. 418-425, 2005.

[12] P. Loncomilla and J. Ruiz-del-Solar, *Robust Object Recognition using Wide Baseline Matching for RoboCup Applications*, in RoboCup 2007: Robot Soccer World Cup XI, LNCS, vol. 5001, pp. 441-448, 2008.

[13] U. Kaufmann et al., *Visual Robot Detection in RoboCup Using Neural Networks*, in D. Nardi et al. (Eds.): RoboCup 2004, LNAI, vol. 3276, pp. 262–273, 2005.

[14] S. Volioti and M.G. Lagoudakis, *Histogram-Based Visual Object Recognition for the 2007 Four-Legged RoboCup League*, in Artificial Intelligence: Theories, Models and Applications, LNCS, vol. 5138, pp. 313-326, 2008.

Large Scale Egomotion and Error Analysis with Visual Features

M. Cazorla, D. Viejo, A. Hernandez, J. Nieto and E. Nebot

***Abstract*—Several works deal with 3D data in SLAM problem but many of them are focused on short scale maps. In this paper, we propose a method that can be used for computing the 6DoF trajectory performed by a robot from the stereo images captured during a large scale trajectory. The method transforms robust 2D features extracted from the reference stereo images to the 3D space. These 3D features are then used for obtaining the correct robot movement. Both Sift and Surf methods for feature extraction have been used. Also, a comparison between our method and the results of the ICP algorithm have been performed. We have also made a study about errors in stereo cameras.**

***Index Terms*—Computer Vision, Mobile Robotics.**

I. Introduction

During the last years, new 3D sensor devices have been developed, and computing capabilities have been improved. These improvements can be used to obtain and process a better robot environment information in the field of mobile robotics [1], [2], [3]. By now, methods for achieving tasks such as localization [4], [5], [6], navigation [7], [8] or automatic map building [9], were restricted to the two dimensional world which could be captured by the robot sensors. Nevertheless, using the new 3D sensors such as stereo cameras or 3D laser range finders it is possible to improve the representation of observed objects in order to use them into applications such as augmented reality, architecture, manufacturing process, SLAM problem, etc. Furthermore, this new dimension can be used to improve the methods and behaviors used by a robot in order to accomplish its objectives. In this way, the robots equipped with this new 3D sensors are able to move freely into a 3D space, without being confined to the ground, watching and avoiding 3D shape and volume objects.

In the mobile robot field, one of the most important issues that has to be considered is the movement performed by the robot between two consecutive poses. This information can be obtained from different sources. The most common solution consists in using the robot internal odometers to estimate its movement. Nevertheless, using odometry causes two main problems. First, odometry information always includes measurement errors, which affect the results. Second, it is possible to work with a robot without odometer sensors, or whose odometry information is quite imprecise. This is a very common situation for robots that are able to perform six degrees of freedom (6DoF) movements into a real 3D environment. For this reason, it is necessary to obtain a movement estimation as accurate as possible. In order to compute this movement estimation, robot surroundings data grabbed by its sensors are used. This kind of solutions for robot movement estimation are known as egomotion or pose registration methods [10], [11], [12].

Using 3D information in order to get the 6DoF transformation from the robot (egomotion) is not an easy task. Although several approaches have been used (ICP [13], [14], Ransac [15], etc.) all those approaches do not work in the presence of outliers (features seen in one frame and not seen in the other). The greater the robot movement the greater the number of outliers are, and the classical methods do not provide good results. In this paper, we propose the use of visual features (like Sift [16], SURF [17]) from the 2D image together with 3D information from stereo processing. We have to deal with the 3D error of the stereo camera: 3D points close to the camera have less error than points far from it.

The paper is organized as follows: First, in section II we present an overview of the physical systems used for obtaining 3D data. Then, Section III presents the method used to get the practical error from the stereo cameras. Later, in section IV we briefly describe the feature extractors used in this study. Our approach for computing egomotion from consecutive stereo images is described in Section V. The results of our experiments are discussed in section VI. In Section VII we briefly analyze some errors present in the results. Finally, we present all the conclusions obtained during this paper and the future work in section VIII. The later is focused on obtaining a more complete 3D model that could be used for estimating robot movements. We also plan to include robot movement estimation into a global error rectification algorithm.

II. Data Acquisition

The data is taken from two different cameras which are mounted on the top of a vehicle, as shown in Fig. 1. One camera is a Bumblebee XB3 from PointGrey with two different baselines, 24cm and 12cm. The other camera comes from Videre and allows different configurations for its baseline. In this case, we have set it to its maximum (60cm) in order to improve the resulting stereo data for further objects. We have acquired a set of images with the vehicle at 30-40km/h. We have driven it in a path through the University of Sydney, with a distance of 3-4kms, in a real scenario with other vehicles and pedestrians around. The frame rate is, approximately, 8Hz

Diego Viejo and Miguel Cazorla are with University of Alicante.
E-mail: {dviejo, miguel}@dccia.ua.es

Andres Hernandez, Juan Nieto and Eduardo Nebot with University of Sydney.
E-mail: aher5442@mail.usyd.edu.au, {j.nieto, nebot}@acfr.usyd.edu.au

and we have processed the sequence offline. There are more than 4000 images in total.

Fig. 1. Vehicle used in the experiments and detail of the cameras placed on it.

III. Error modelling

We are interested in evaluating the measurement error in depth provided by our stereo cameras. We know the theoretical errors that are provided by the manufacturer, but the real error seems to be higher than those. We have to be sure about the error, so we are going to measure it. To do that, we use the system shown in Figure 2. We have placed the two stereo cameras that we are going to use in our experiments together with a laser Sick. As the error of the laser is negligible compared to the error of the cameras, it provides a good "ground truth" for computing camera's errors. We place a plane surface in front of the cameras and take a complete reading of the laser and the two cameras at the same time. The measurement is done from 1 to 30 meter, with intervals of less than a meter. As shown in Figure 3, the theoretical error is much less (usually three times less) than the real error. We'll use this error information in order to improve the accuracy of the method described in the next sections.

Fig. 2. Error modelling system set up. A 2D laser is used together with our stereo cameras. The information obtained by this laser is used as "ground truth" for computing the stereo cameras real measurement error.

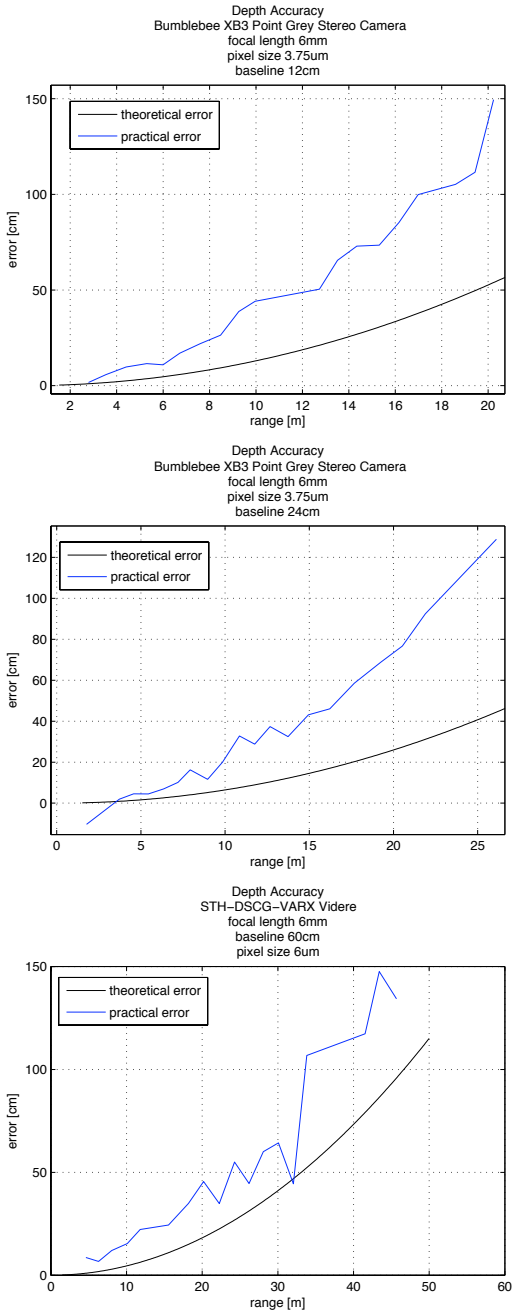


Fig. 3. Measurement error analysis. These graphics show the relation between the distance at which an object is observed and the measurement error obtained. The first two graphics come from the Bumblebee XB3 stereo camera for 12 and 24 cm. baseline configuration. The third graphic shows the error information from the Videre camera configured with a 60 cm. baseline. The real error (in blue) is compared to the theoretical error (in black) provided by the manufacturer.

IV. Visual Features

In this section, we briefly describe the two features used and compared in the paper. One of the most used visual feature is SIFT [16], a method used in computer vision to detect and describe features in an image. It performs a local pixel appearance analysis at different scales. The SIFT features are designed to be invariant to image scale and rotation. Furthermore, it obtains a descriptor for each feature that can be used for different tasks such as object recognition. SIFT algorithm is divided into two main parts. In the first one, the location of the points of interest is extracted. The image is convolved using a Gaussian filter at different standard deviations σ. Then, the difference of Gaussians (DoG) is computed as the difference between two consecutive Gaussian-convolved images. This process is repeated in order to obtain the DoG for the input image at different scales. The localization of the points of

interest starts when all DoG have been computed. A point located in a DoG is considered as a point interest if it has the maximum/minimum value compared with its 8-neighbours in the same DoG and with the 9-neighbours in the adjacent DoG at superior and inferior scale. The localization of the points of interest is then improved by interpolating nearby data, discarding low-contrast points and eliminating the edge responses. In the second part of the algorithm a descriptor vector is computed for each point of interest. Based on the image gradient around a point of interest an orientation for this point is computed. This orientation represents the starting point from where the descriptor array is computed. This is a 128-element array that holds the information about 16 histograms of 8 bins computed from the same gradient data. Finally, this descriptor vector is normalized in order to enhance invariance to changes in illumination.

On the other hand, we have analyzed the Speeded Up Robust Features (SURF [17]). This method is partly inspired by the SIFT descriptor. Instead of modifying the scale of input image, SURF algorithm uses a concept known as integral images that allows scaling up the image filter in a constant time. Furthermore, the SURF descriptor is held in a 64-element array instead of a 128 one. In order to compute this descriptor, instead of the gradient the first order Haar Wavelet distribution is used. Having the same rotation and scale invariance response, SURF is faster than SIFT in finding features and its descriptors. Furthermore, since the descriptor size is also lower, SURF-matching methods are also faster than the SIFT-based ones. Despite the speed, both kinds of feature extractors provide similar results for matching applications.

V. Egomotion

In order to get the egomotion of the vehicle, we present an ICP-like method to match 3D features. First, we process the base image (the image which is centered in the coordinate system of the camera) to get features (Sift and Surf in this paper, but any 2D feature can be used). Then, we match the 2D coordinates (image coordinates) of the feature with the 3D data from the stereo processing, in order to provide the feature with 3D coordinates. Fortunately, stereo cameras provide information about which 2D coordinate is associated with 3D data. Thus, matching is very simple: we look for the 3D data of a 2D coordinate (the center of a 2D feature). If there is this information, we give the 3D coordinates to the 2D feature). If not found due to, for example, lack of texture, we search around the 2D point in a certain neighborhood and provide a mean value of the 3D data found in this neighborhood.

Due to the errors presented in Section III, we have decided to select features close to the camera, because the longer the distance to the camera, the greater the 3D error. Thus, only features with a Z distance below a threshold are selected to match between two consecutive sets. We have to take into account the movement between two consecutive frames, in order to select a set of features in both frames which intersect and have enough number of features to match. If movements are limited to, for example, 1 meter we select features from 1 to 2 meters in the first frame and from 0 to 1 in the second one. If there are not enough matches, we expand the limits from 1 to 3 meters and from 0 to 2, and so on to find a minimal number of matches or to reach a long distance (10 or 20 meters, depending on the baseline).

Once we have found matches between two consecutive frames, we apply an ICP-like algorithm to find the 3D transformation between frames. ICP is a classical algorithm to match two 3D point sets, but it can not find a good alignment in the presence of outliers. For long movements, ICP does not give good results, because there are a lot of outliers. In our case, where movements are from 0.5 to 4 meters, ICP is not a good approach. But using features like Sift or Surf, with an additional information, i.e. descriptors which are robust to brightness change and point of view change, are good enough for this task. So we use descriptors to find matches, instead of using Euclidean distance like the original ICP.

We have found several problems in our approach. We selected only the closest points in order to avoid the error (Z error) from the furthest points. The distance mean error (X, Y, Z) between two consecutive frames is below 1cm, but the angle error is not acceptable. This angular error comes from the fact that small angular errors in closest points yield bigger angular mean errors. We have used closest points to get the translation term (Tx, Ty, Tz) and all the points for angular term (Rx, Ry, Rz). In Figure 4 we show some results. In this figure we have used the Sift features.

VI. Results

The result of applying the proposed method to our data set, taken at the University of Sydney, is shown in Figure 5. It corresponds to the first part of the sequence. Three different resulting trajectories are shown. In two of them the proposed method was applied using Sift and Surf features and descriptors. Red and blue lines correspond to the trajectories computed in this way. The green line was obtained by applying ICP algorithm directly over each two consecutive 3D stereo sets. The ICP result is used to compare the results of our proposal with a classical 3D egomotion approach. Sift based egomotion seems to provide the best results, compared to Surf's and the ICP's, as it fits almost correctly the real trajectory followed by the robot.

Once we have computed a trajectory using the method proposed in this paper, we can use it in order to represent all the observations within a common reference system. This reconstruction makes up a 3D map of the environment of the robot. For the experiment explained before, Figure 6 shows the resulting 3D map for the trajectory obtained from the University of Sydney with the Sift based approach explained in this paper. It is a free 3D view of the computed map. Robot trajectory is represented with a red line. All the 3D points from the scenes taken by the robot at each pose are stored together using an occupancy grid. This kind of maps can be used for 3D localization or volumetric obstacle avoidance in 3D. The trajectories obtained with the other methods (Surf and ICP) don't provide clear results due to the error committed and therefore are not shown.

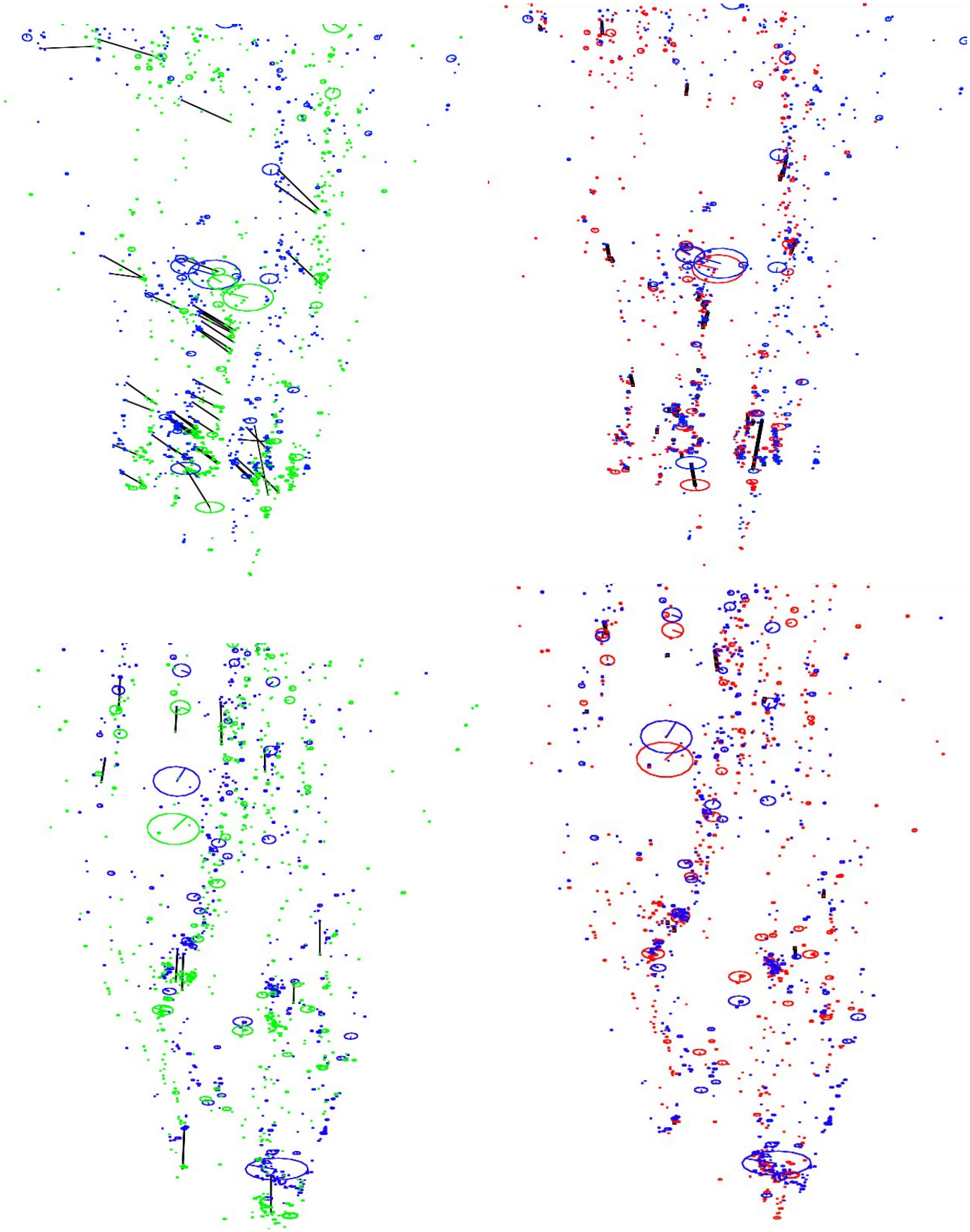


Fig. 4. Two examples of the matching method. Left: initial matching (without applying any algorithm) Right: final matching, result of applying our method). The black lines indicate matches. Top: first experiment: only 51 out of 850 possible matches were used. The distance error was 2,5mm and the execution time was 169ms. Bottom: second experiment: only 22 out of 860 matches were used. Distance error of 6,2mm and execution time of 144ms.

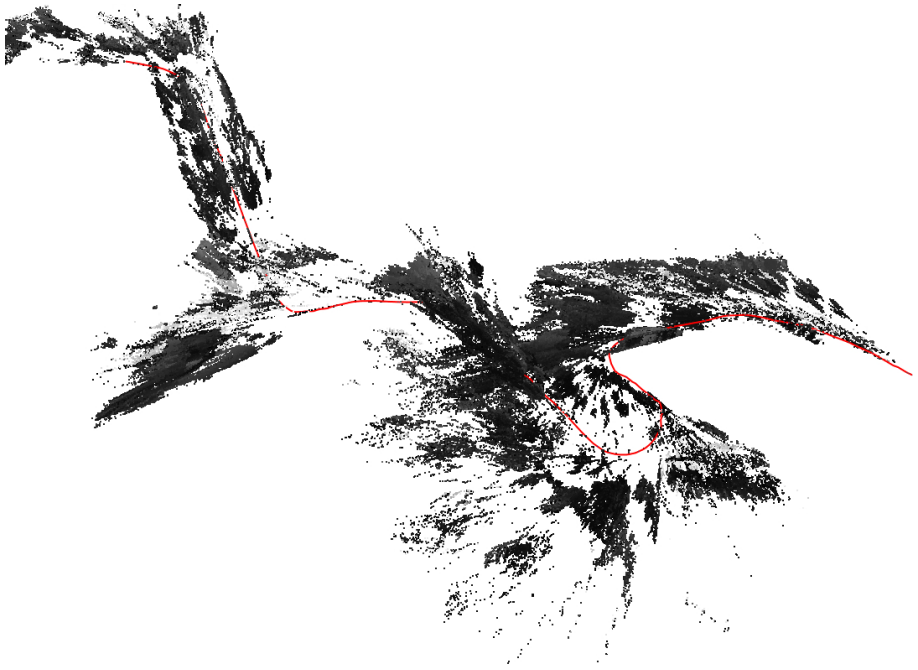


Fig. 6. Reconstructed 3D map from computed trajectory (red line). All the 3D scenes observed by the robot during its trajectory are stored in a reference frame.

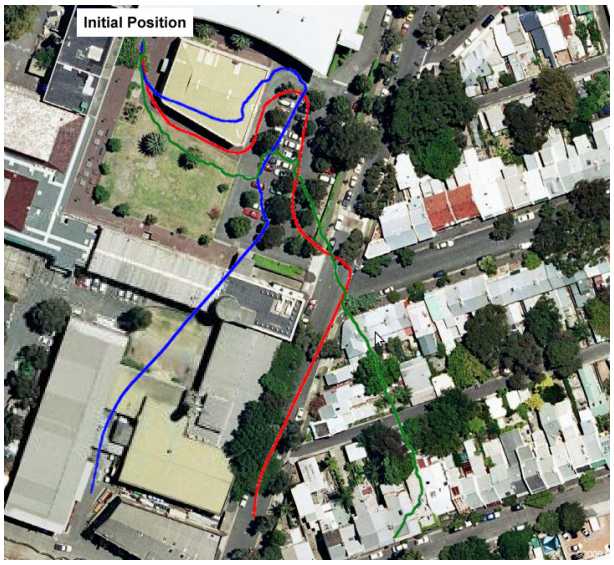


Fig. 5. Results from the experiment carried out at the University of Sydney. Three different trajectories are placed on top of an aerial picture of the environment. The result of applying our proposed method with Sift features is represented by the red line, Surf based result is represented in blue and ICP egomotion algorithm result is shown in green.

VII. Large Scale Errors

The previous method presents a good egomotion estimation in small paths. However, there still is a residual error, which makes the final trajectory unreadable after 100 meters. It comes from the fact that we use a least squares approach to find the transformation between two consecutive frames. However, the least square method is only valid if errors are isotropic (see Figure 7). If not, a different (and not so obvious) approach must be followed. When using a 3D laser in large paths, errors are very low compared with stereo. So, if the previous method is applied in conjunction with 3D laser data, the results could be acceptable. However, with stereo data [18] this error produce a reconstruction like the one shown in Figure 9. The transformation found at the egomotion step has a rotation error (around X axis) and some minor translation errors. This provides an additive error which produces a hill where an almost plane path must be. In very large paths we have detected a "tornado-like" effect, with loops around the X axis.

Another source of error is present when our vehicle is stopped, like in a traffic light. If another vehicle is in front of us, and as this is closer than other objects, our algorithm uses 3D information from this vehicle, which is moving like our vehicle, with the result of a bad egomotion. A better study of this situation must be done.

VIII. Conclusion

In this paper we have presented a method for large scale robot 6DoF movement estimation using stereo images. We have also performed a comparison between Sift and Surf feature extractors. Although Sift is slower than Surf, it provides better results for our method. It is not clear where these bad results come from when using Surf, since previous works show how both methods provide almost the same results. In this way, a deeper study must be done. We have also demonstrated how the trajectory computed with our method improves the results obtained with the classical ICP algorithm, commonly used for computing 6DoF egomotion for two 3D set of points.

On the other hand, we have made a study about errors in this framework. First, real errors from stereo system have been calculated, to be used in the matching process. Second, some

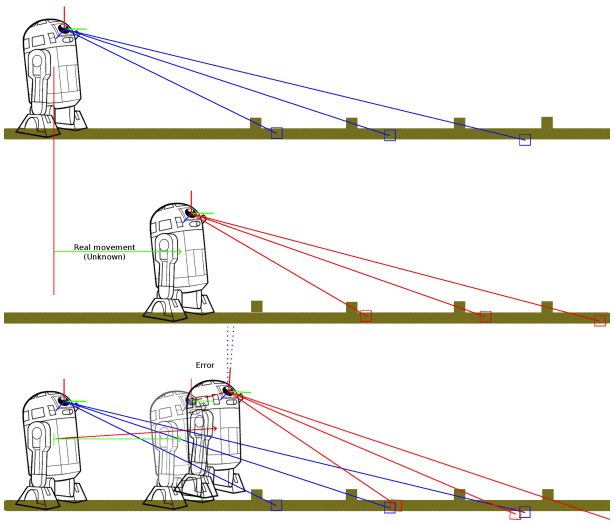


Fig. 7. Pose error due to anisotropic error. Least square method tries to fit features with different errors and the obtained result has an error of rotation (and some translation), which is accumulated.

Fig. 8. Cenithal view of the reconstructed trajectory. This result is not so bad, but in large distances (like the end of the path, to the right) a straight path is obtained as a curve.

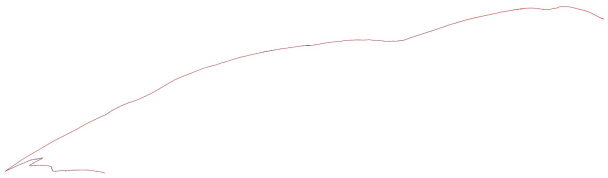


Fig. 9. Robot View of the reconstructed trajectory. The trajectory should be almost a horizontal line (without elevation) in this part, but due to the accumulative errors (see Figure 7) is like a hill.

situations which must be taken into account in real scenarios have been described.

As future work, we plan to minimize the effect of anisotropic errors by reprojecting 3d locations of the matched features on the image plane. Then we can use the most consistent euclidean distance between features in the image plane instead of 3D euclidean distance. We want also to merge information from both cameras in order to improve the final result.

Besides, we are planning to model dynamic objects in the environment (like pedestrians, other vehicles and so on), because that can yield (and effectively does) a great source of errors.

ACKNOWLEDGMENT

This work has been supported by grant JC08-00077 from Ministerio de Ciencia e Innovación of the Spanish Government. We would like to thank Rio Tinto Mine Automation Center for their support in this work.

REFERENCES

[1] J. Weingarten, G. Gruener, and R. Siegwart, "A Fast and Robust 3D Feature Extraction Algorithm for Structured Environment Reconstruction," in *None*, 2003.

[2] D. Schroter and M. Beetz, "Acquiring models of rectangular 3d objects for robot maps," in *Proc. IEEE International Conference on Robotics and Automation ICRA '04*, vol. 4, pp. 3759–3764, Apr 26–May 1, 2004.

[3] C.-T. Kuo and S.-C. Cheng, "3d model retrieval using principal plane analysis and dynamic programming," *Pattern Recogn.*, vol. 40, no. 2, pp. 742–755, 2007.

[4] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," Tech. Rep. IAI-TR-97-3, 25, 1997.

[5] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[6] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[7] A. Tsalatsanis, K. Valavanis, and N. Tsourveloudis, "Mobile robot navigation using sonar and range measurements from uncalibrated cameras," in *Proc. 14th Mediterranean Conference on Control and Automation MED '06*, pp. 1–7, June 2006.

[8] G. L. Mariottini and D. Prattichizzo, "Image-based visual servoing with central catadioptric cameras," *Int. J. Rob. Res.*, vol. 27, no. 1, pp. 41–56, 2008.

[9] S. Thrun, D. Hahnel, D. Ferguson, M. Montemerlo, R. Triebel, W. Burgard, C. Baker, Z. Omohundro, S. Thayer, and W. Whittaker, "A system for volumetric robotic mapping of abandoned mines," in *Proc. IEEE International Conference on Robotics and Automation ICRA '03*, vol. 3, pp. 4270–4275, 14–19 Sept. 2003.

[10] M. Agrawal, "A lie algebraic approach for consistent pose registration for general euclidean motion," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1891–1897, Oct. 2006.

[11] O. Koch and S. Teller, "Wide-area egomotion estimation from known 3d structure," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR '07*, pp. 1–8, 17–22 June 2007.

[12] R. Goecke, A. Asthana, N. Pettersson, and L. Petersson, "Visual vehicle egomotion estimation using the fourier-mellin transform," in *Proc. IEEE Intelligent Vehicles Symposium*, pp. 450–455, 13–15 June 2007.

[13] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *1991 Proceedings., IEEE International Conference on Robotics and Automation, 1991.* (G. Medioni, ed.), pp. 2724–2729 vol.3, 1991.

[14] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[15] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[17] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[18] G. Dubbelman and F. Groen, "Bias reduction for stereo based motion estimation with applications to large scale visual odometry," in *Proc. of Computer Vision and Pattern Recognition*, pp. 2222–2229, 2009.

Mission Specification in Underwater Robotics

Enrique Fernández-Perdomo, Jorge Cabrera-Gómez,
Antonio C. Domínguez-Brito and Daniel Hernández-Sosa

***Abstract*—This paper describes the utilization of software design patterns and plan-based mission specification in the definition of AUVs missions. Within this approach, a mission is described in terms of a set of task-oriented plans in order to simplify mission definition and favor reutilization of some aspects of a mission. Each plan organizes how and when basic tasks like measurement sampling, navigation or communication are to be carried out. The usage of design patterns for AUVs has been considered in order to ease system architecture design.**

***Index Terms*—Software Engineering, control architecture, underwater robotics, mission, framework.**

I. Introduction

DURING the last ten years, the adaptation of Software Engineering principles and methodologies to robotics has caught a lot attention in this field [8]. Nowadays, it is widely accepted that the final quality of a sensory-motor system, its cost of development and implementation, and its ease of usage, are highly conditioned by the software engineering methodologies and tools utilized. In this sense, concepts like design-for-reuse or the utilization of design patterns [10], components and programming frameworks [8] are now routinely used in the development of complex robotic systems.

Autonomous underwater vehicles (AUV) are valuable and highly sophisticated robotic devices and their programming is a complex task that demands an important effort from all the engineers, programmers and scientists involved. The complexity of these systems increases as new mission scenarios are proposed requiring —for example— larger periods of autonomy or coordination among multiple vehicles.

A key aspect of AUVs programming is that concerned with mission specification and control [7]. In particular, special attention must be paid to the number of intervening actors involved in the specification and development of a AUV mission: from system software architects and device integrators to scientists acting as final users of the vehicle.

Mission plans are to be written by scientists, who —probably— will not be experts in robotics, so it is important that a mission can be easily expressed in terms of basic activities of navigation (go to a point, achieve a depth, ...), measurement (take a measure with certain instrument), logging data, and communication. Due to the same reason, mission plans should be analyzed and validated automatically beforehand. Their expression must be concise and favor the reuse and adaptation of basic mission plans to new contexts. In particular, a clear separation between the software that is in charge of the AUV's control and navigation and the code that contains the mission specification is highly desirable [1]. In-situ changes of mission plans should be possible.

As commented previously, AUVs must exhibit large periods of true autonomy so they must be programmed to be reliable. The definition of procedures to detect anomalies or possible malfunction, and to recover from unexpected errors or exceptions is obliged. Most of the exception handling will be integrated within the vehicle control system and should be reusable between missions, but it should also be possible to redefine the treatment of certain failures in the context of a specific mission.

In this paper, the main elements of a software framework for programming AUVs will be presented. The section II will be devoted to present an approach for mission specification based on a set of plans, each of them conceived to cover the different facets of AUV's activities (communications, navigation, measurement, etc.). In section III, we describe the usage of design patterns within a component-based framework for building the software control architecture. Final sections include a review of other related works and a final summary.

II. Mission Specification

A mission is defined as the set of tasks a vehicle must perform. From most common AUV tasks analysis, a basic and proper mission specification framework can be outlined. It is possible then to evaluate qualitatively the features of a particular mission specification design depending on a series of parameters such as modularity, flexibility, monitoring, ease of definition and others, which are key features through mission life cycle stages.

A. Typical AUV tasks

In general, typical tasks included in almost every AUV mission can be orthogonally enumerated as:

1) *Measure sampling:* Sensory equipment is managed to sample different measures, which may be saved or transferred to surface station.
2) *Path following:* Given a set of waypoints, the vehicle tries to reach them in sequence. Therefore, it follows the path obeying some motion constraints.
3) *Area exploration:* The vehicle moves inside an area describing a particular pattern —e.g. *zig-zag*, concentric, spiral, etc.— to explore it.

All authors are members of the University Institute SIANI (Intelligent Systems and Numeric Applications in Engineering — Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería) at University of Las Palmas de Gran Canaria (ULPGC). E-mail: { efernandez, jcabrera, adominguez, dhernandez}@iusiani.ulpgc.es

This work has been partially supported by the following research projects: Project *PI2007/039* funded by the Autonomous Government of Canary Islands (Gobierno de Canarias — Consejería de Educación, Cultura y Deportes, Spain) with FEDER funding; and Project *TIN2008-06068* funded by the Ministerio de Ciencia e Investigación, Gobierno de España.

4) *Measure tracking:* The vehicle senses a particular measure and tries to track it using a predefined behavior.
5) *Communication:* Two main communication tasks are noticed:
   a) *Data/Mission transferring:* Send or receive single or multiple measure samples or internal system observable variables.
   b) *Coordination:* A network of vehicles communicate among them to achieve common goals.

Other vehicles like buoys and ships —e.g. ASVs (Autonomous Surface Vehicles)— may take advantage of some of these tasks, since AUVs capabilities cover most oceanographic missions. Additionally, these elementary tasks might be combined frequently.

B. Mission life cycle

The mission life cycle is composed of the different stages that take place during mission specification, from creation to execution finalization in the vehicle. Diagram in Fig. 1 represents the typical tasks involved in sequence.

1) *Creation:* Process of mission creation or edition. It is common to have a planning GUI (Graphical User Interface) to aid in the specification concerns. Independently of the availability of such a planning tool, an internal textual representation is mandatory to accomplish the next stages of the mission life cycle. Common representation approaches are DSLs (Domain Specific Languages) and Petri Nets coded with languages like Prolog or Lisp [1], [19], [20], [7].
2) *Validation:* It is possible to apply a validation process on a mission completely defined. This will usually be an automatic and transparent process embedded in a verification tool, which will manage the mission textual representation to detect errors or incompatibilities within mission elements.
   Depending on the mission design and representation, the validation techniques vary —e.g. Petri Nets formalism allows network correctness verification [16], [5], [7], a DSL designed with XML (eXtensible Markup Language) can be automatically validated if XML Schemata are provided, etc. Additionally, further semantic checks can be applied.
3) *Transference:* The former stages take place on a computing environment outside the vehicle. Once validated, the mission is transferred to the vehicle, which will save it. This is not trivial tough, because verbose mission representations may consume excessive bandwidth and power. Furthermore, mission modifications may be sent while mission execution as part of replanning.
4) *Execution:* Once the mission is saved in the vehicle, execution will begin after the starting order has been received. In general, we can identify three internal stages:
   a) *Load:* When vehicle's system is booted, mission must be loaded into memory using a data model that describes the mission in order to be interpreted and managed —e.g. Petri Nets are interpreted directly, DSL representations must be parsed and used to build a data model, etc.
   b) *Interpretation:* System is configured according with mission specification while it is interpreted. If any error or exception occurs, the system will log it and try to act accordingly.
   c) *Finalization:* Mission is ended when all specifications have been carried out. A deconfiguration process leaves the vehicle idle, i.e. releases resources, turns devices off, etc.
5) *Replay/Simulation:* Though not compulsory, a simulator may load information logged while mission execution and replay or recreate what happened, allowing offline mission analysis. Predicted or approximate environment conditions provided, simulation can be done before real mission too, allowing beforehand non-predictable incompatibilities detection.

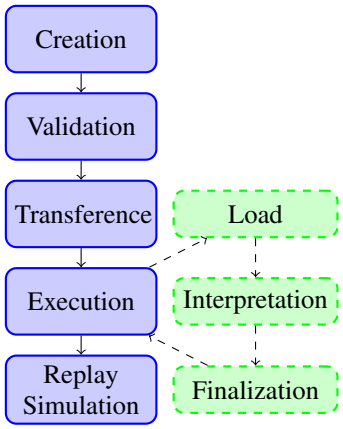


Figure 1: Mission life cycle stages

Despite mission specification modularity, there may still appear dependencies, so the validation stage is necessary to detect inconsistencies or incompatibilities. An incompatibility is an impediment to accomplish a task specified in the mission, usually caused by dependencies within mission elements, available vehicle equipment or vehicle's embedded system state. There are two types of incompatibilities:

1) *Predictable:* Incompatibility can be detected analyzing mission specification. A validation engine integrated in a planning tool usually achieves this task.
2) *Non-Predictable:* Incompatibility cannot be detected with a validation process. This kind of incompatibility arises during mission execution. Vehicle's embedded system is responsible for detecting them and acting accordingly —e.g. replanning, task priorization, exception throw.

C. Plan-based missions

As Fig. 2 shows, mission is split in several plans minimizing dependencies between them and enabling plan exchange among different missions. Anyway, integrity checks between plans can be tackled with a mission validation process.

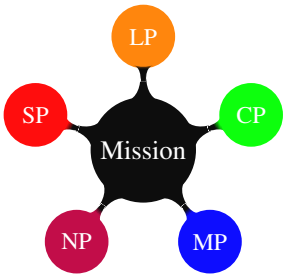


Figure 2: Mission Plans

Furthermore, it is desirable to have a high level mission specification independent of AUV control software.

A plan can be defined as a block that specifies a set of similar and related tasks from a particular field. Following plans are proposed to cover common AUV missions:

1) *Logging Plan (LP):* Logging system or *Black Box* configuration that indicates data logging tasks and stock management. Common *log-able* data are measures and system observable variables.
2) *Navigation Plan (NP):* Details navigation tasks that state where the vehicle must go and forbidden areas. Typical navigation missions are allowed, such as path following, area exploration and measure tracking, by means of actions with maneuver semantics.
3) *Communication Plan (CP):* Contains communication tasks, which are limited by vehicle's communication equipment. Mission validation allows premature detection of incompatibilities between CP and available equipment or other plans —e.g. NP may constraint communication equipment coverage and bandwidth.
4) *Measurement Plan (MP):* Specifies measurement tasks regarding data provided by available vehicle's equipment, with optional sensor selection.
5) *Supervision Plan (SP):* Declares tasks to carry out in case of execution fault. It fires any kind of action through commands or contingency plans.

Plan-based missions are basically inspired on sub-goals and tasks mission specifications, but additional features are considered:

1) Plans gather tasks from a particular typology. Modular specification allows plan reuse and exchange.
2) Plan-based missions far from just provide modular specifications, also bring modular interpretation and management. Vehicle's system architecture may profit from this fact —e.g. each plan interpretation may be accomplished by a dedicated subsystem. Therefore, besides control from mission specification separation, plans also allow system to manage each plan separately in different subsystems that may act like internal agents modeled as software components that coordinate to perform the mission, resolve conflicts and understand a common control language.
3) XML is used as mission specification language, with the following benefits:
   a) Human readable, with semantic close to domain represented.
   b) Great variety of software tools. Programming time is reduced and specification, validation and interpretation portability is increased —e.g. parsers, XML validators, etc.
   c) Language formal syntax is encapsulated within XML Schemata that allow mission validation.
4) Configurable parameters aid mission elements modification during execution. XPath query language is employed to specify which mission elements alter. Mission parameters are simply XPath queries shortcuts, hence actually any mission element is accessible directly.
5) GUI tools complement mission life cycle management, specially by means of assisted creation and automatic validation. This relaxes vehicle's system from further checks during mission load and interpretation.

Plan-based mission specification is analyzed according with the following criteria:

1) *Modularity:* Plans are mission specification modules, internally described in terms of elementary mission tasks. Depending on the plan, tasks may be taken in sequence or parallelly —e.g. NP paths and areas are usually specified on a sequential basis.
2) *Flexibility:* Tasks are designed to cover all typical AUV missions. Mission configuration parameters allow dynamic mission modifications.
3) *Monitoring:* Task level monitoring supported by sub-goals and tasks mission specification is extended to plans.
4) *Ease of definition:* Mission definition is textually represented using XML. Although XML facilitates mission definition, the use of assisted GUI tools are encouraged for end users. Furthermore, tasks abstraction level is close to AUV mission semantic.
5) *Portability:* Each mission plan has an XML Schema to take advantage of validation and interpretation tools, which are portable and widely used.
6) *Reconfiguration:* Mission parameters are the basic support for mission reconfiguration. XPath is used to access mission elements directly or through configuration parameters names.

D. Plan specification

Every plan is made up of a task list, each task with name and unique identifier within the plan. Task specification contains the elements below:

1) *Triggers list:* Specifies the conditions that activate the task. Triggers are predicates on measures or system variables. When all triggers are activated a notification signal is generated, forcing execution of all actions. If empty, actions will have to run continuously.
2) *Actions list:* Specifies name and parameters of commands to execute under system supervision and monitoring while triggers are activated. Actions are system primitives or *reactive skills* that represent atomic or elemental capabilities as in RAP system [9].
3) *Inhibition period:* Inhibits trigger's state checking to ensure task execution is kept at least for a given time,

even if triggers get immediately deactivated. Triggers are updated asynchronously and sample-driven. Once triggers get activated they are only checked periodically though, until they get deactivated again.

Three types of triggers are proposed:

1) *Condition:* Comparison between a particular measure sample and a given value applying a relational operator. If comparison result is true, it is triggered.
2) *Interval:* Temporal interval or range of values built from a given tuple of initial, increment/period and final values. If a particular measure sample is inside the interval, it is triggered.
3) *Exception:* System exception name. If such exception is thrown, it is triggered.

Being predicates, triggers may be combined logically. Triggers list is logic-and combined, while logic-or can be expressed using several tasks with the same actions list. Therefore, descriptive specification instead of procedural or sequential is basically supported by means of trigger-driven tasks. List. 1 shows an example where two tasks are defined to indicate turbidity sampling above 100m deep, and salinity and temperature continuously.

```
<mp id="1" name="offshore salinity-turbidity">
  <task id="1" name="surface turbidity">
    <condition measure="depth" operator="le"
               value="100" unit="m"/>
    <sample measure="turbidity">
      <frequency value="1" unit="Hz"/>
      <resolution value="0.1" unit="NTU"/>
    </sample>
    <inhibition value="1" unit="min"/>
  </task>
  <task id="2" name="salinity profile">
    <sample measure="salinity">
      <frequency value="0.1" unit="Hz"/>
      <resolution value="0.01" unit="PSU"/>
    </sample>
    <sample measure="temperature">
      <frequency value="0.1" unit="Hz"/>
      <resolution value="0.1" unit="C"/>
    </sample>
  </task>
</mp>
```

Listing 1: Measurement Plan example

List. 2 shows a basic NP example that contains a forbidden area and three navigation tasks: initial path following, area exploration and final path following. This plan specification may be generated automatically by a planning tool that allows a graphical specification like depicted in Fig. 3.

```
<mp id="1" name="bay exploration">
  <task id="1" name="go to bay">
    <waypoint id="1">
      <position x="9" y="17" z="0" unit="km"/>
    </waypoint>
    <waypoint id="2">
      <position x="10.5" y="16.5" z="0" unit="km"/>
      <aptitude roll="0" pitch="-5" yaw="90"
                unit="degree"/>
    </waypoint>
    <transect id="1" start="1" end="2">
      <speed x="5" y="0" z="0" unit="m*s^-1"/>
    </transect>
  </task>
  <task id="2" name="bay exploration">
    <area id="1" name="bay">
      <depth min="0" max="150" unit="m"/>
      <time value="2" unit="h" margin="10">
      <path id="1">
        <transects mode="zigzag" amount="10">
          <time value="10" unit="min"/>
          <depth value="10" unit="m"/>
          <angle value="60" unit="degree"/>
        </transects>
      </path>
    </area>
  </task>
  <task id="3" name="come back port">
    <waypoint id="1">
      <position x="9.5" y="14.8" z="0" unit="km"/>
      <uncertainty value="1" unit="m"/>
    </waypoint>
  </task>
  <forbidden id="4"/>
</mp>
```

Listing 2: Navigation Plan example

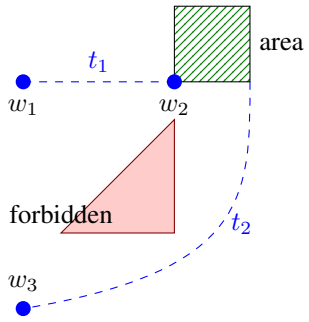


Figure 3: Navigation Plan example

Vehicle's system is configured according with all task triggers of all plans. Measures used by triggers are provided by measurement actions internally and automatically requested. This allows trigger evaluation and signaling to execute task actions in a trigger-driven basis. Fig. 4 shows how *Plan Interpreter* and *Trigger Dispatcher* configure the system (- - -) to enable task activation signaling (——).

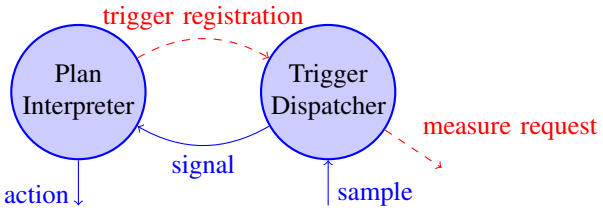


Figure 4: Trigger-driven task issue

Action execution is actually a system configuration process, while action finalization consists in reverting it, i.e. task's actions semantic is activation. Additionally, tasks inside a single plan are only allowed to use a particular set of actions, which are those related with plan semantics. On action failure basic ignore, retry or abort behavior may be selected for each possible action exception. More elaborated exception handlers may be specified with tasks contained in the *Supervision Plan*, which is responsible for mission state monitoring, exception resolution and coordination of system elements.

III. Architectural Design Patterns

In order to design a consistent and integrated system, the underlying software technology supporting the execution

of plans and tasks has to be selected accordingly to high-level objectives. Our election combines the use of a generic component-based programming framework with design patterns for robotic software. On one side, the framework contributes to reduce the programming effort promoting modular and robust reusable code. On the other side, design patterns contribute to improve software quality.

A. Frameworks and Design Patterns

For developing the architecture we are introducing in this document we will take an approach already used in other areas of robotics, where some programming frameworks have come out in order to provide solutions to some of the recurrent problems faced when building control software for robotic systems. Those frameworks have blossomed in many areas, from service and edutainment robotics to space applications [13], [23], [11], [17]. A good detailed survey can be found in [8]. In particular, we will make use of a component-oriented programming framework termed *CoolBOT* [6] developed at our lab in the last years. This framework allows building systems by integrating "off-the-shelf" software components following a port automata model [25] that fosters controllability and observability.

Moreover, some design patterns have been identified as quite useful in order to apply them in robotic software control designs [3]. Here, we will consider design patterns as they are proposed in [10]. Thus, a design pattern can be seen as a design solution for a specific problem which can be applied wherever and whenever that solution is valid considering its prerequisites, requirements, and the consequences at system design level of its utilization. Concretely, we will use three basic design patterns adapted from [3]:

- a *Estimation Pattern*,
- a *Control Pattern*,
- and a combination of the former two, a *Reactive Closed-Loop Control Pattern*.

In next paragraphs we will explain those patterns in more detail in order to reinterpret them in terms of *CoolBOT* components.

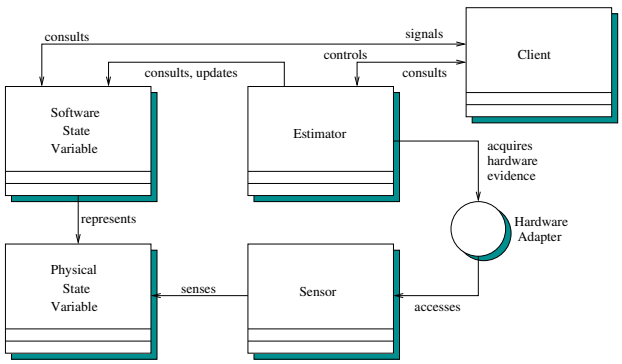


Figure 5: Estimator Pattern.

In Fig. 5 we can observe the architectural structure of the *Estimator Pattern* using a UML object-oriented graphical notation. The distinct entities taking part in the pattern have been represented as objects, independently if they are physically real entities, or just software entities of the control system. This pattern is used for abstracting physical state variables from their counterparts in the control system, establishing a clear separation between the control system and the system under control. Usually, a physical state variable has associated an *Estimator* for estimating the software state variable which represents it in the control system. The *Client* represents an entity of higher level of abstraction which makes use of the pattern. As we can observe, sensory hardware is represented by a *Sensor* object which is accessed through a *Hardware Adapter* which normalizes its interface.

Fig. 6 shows the UML class diagram of the *Control Pattern*. This pattern puts also into practice the principle of separation aforementioned between the control system and the system under control. In this case, the *Controller* makes use of a software state variable in order to actuate and exert actions to control an aspect on the real system, a physical state variable which is its counterpart in the physical system. Similarly to the previous pattern, the *Client* object represents an entity of a higher abstraction level in the control system that makes use of the pattern.

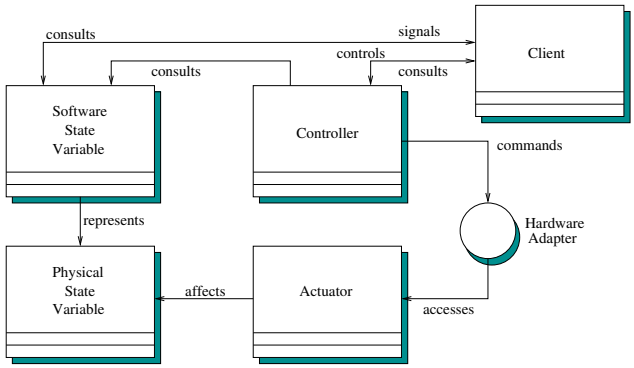
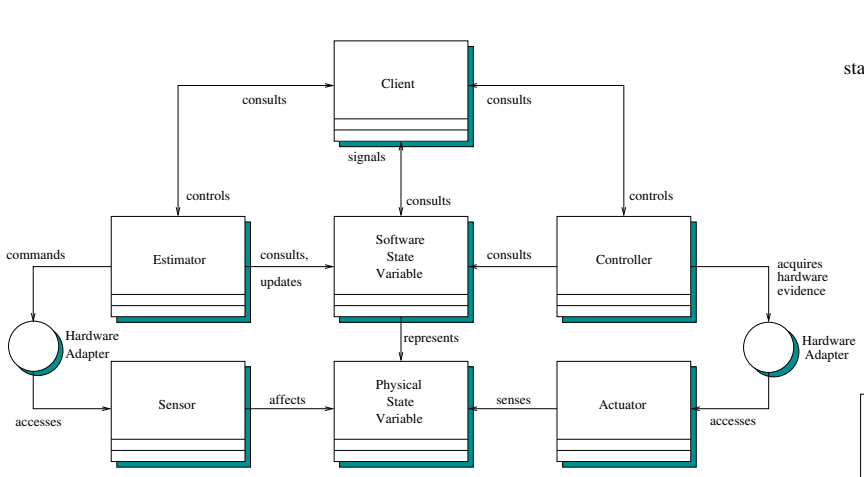


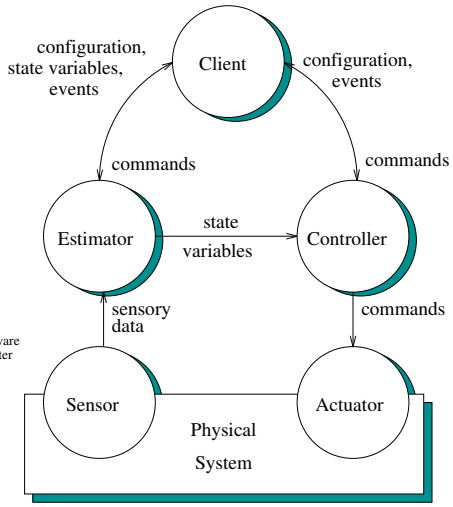
Figure 6: Control Pattern.

The graphical representation of the *Reactive Closed-Loop Control Pattern* is depicted in Fig. 7(a). This pattern is a combination of the previous two, and as a consequence, it is also based on the same principle of control and system separation. Analogously to the previous patterns, there is a *Client* which uses the pattern, and equally, accessing sensors and actuators is normalized using interfaces represented as *Hardware Adapters*. Take into account that in a real control system these patterns are applied indistinctly for any combination of physical-software state variables which are estimated and/or controlled, existing also the possibility of interleaving different state variables in the same reactive closed-loop control pattern. Moreover, many control loops can be operating at the same time in a given moment during system execution.

What is meaningful now for our discussion is how to reinterpret these very well established design patterns in terms of the abstractions and resources available in the software framework we have chosen for implementing our system. In Fig. 7(b) we can observe a representation of the *Reactive Closed-Loop Control Pattern* reinterpreted as a *network* of *CoolBOT* components, where we can distinguish clearly the same entities



(a) Reactive Closed-Loop Control Pattern.



(b) Reactive Closed-Loop Control Pattern in *CoolBOT*.

Figure 7: Reactive Closed-Loop Control Pattern. The design pattern in (a). Translation into *CoolBOT* in (b).

which appears in its corresponding UML representation of Fig. 7(a). With the translation of this design pattern we have tried to map the pattern into *CoolBOT* abstractions, which are usually more complex entities than the objects we frequently find in UML graphical notations representing design patterns. Also notice that entities pertaining to the physical system has not been represented in the figure, contrarily to the UML representations we have used for introducing the design patterns.

In *CoolBOT* a system can be seem as a *network* of *data flow machines* (components) interconnected by data paths (port connections), as we can observe in Fig. 7(b). Port connections among components indicate how data flows within the system, and in this case, how data flows in the pattern, where we can distinguish clearly the control loops present in a system. Notice also that, at each port connection end, and depending on the port connection typology [6], there are some memory buffers storing the information with is under publication on each component. Thus, for example, the *Estimator* component in the figure publishes the software state variables it estimates through its output connections. This explains why in the figure there is no component representing software state variables. It is important also to take into account that UML representations of design patterns, as usually depicted, are mainly static representations of modular designs. In contrast, a configuration of *CoolBOT* components also gives information about the dynamic behavior of the system, because each component is implicitly an *active* data flow machine having its own flow of execution, at least a *thread* in the underlying operating system.

It is important to emphasized here that the design pattern, expressed as a *network* of *CoolBOT* components in Fig. 7(b), is a simplified representation in order to clarify its design. In a real system this pattern may be instantiated multiple times involving multiple instances indistinctly of the different components that integrate the pattern, and where several clients may be sharing and using the services of distinct patterns. Thus, we can outline a system architecture similar to the one shown in Fig. 8 where the lower level, the *Functional Layer*, is mainly composed by multiple reactive closed-loop control patterns which may be indistinctly active/inactive along system's lifetime. Components in higher levels of abstraction make use of the services provided by the system lower level in order to carry out their own services. The intermediate level, *Executive Layer* is mainly integrated by *event dispatcher* and *plan interpreter* components. The last level, the *Supervisor Layer*, is composed by several supervisors that monitor aspects like exception management, system performance, teleoperation, reconfiguration, etc.

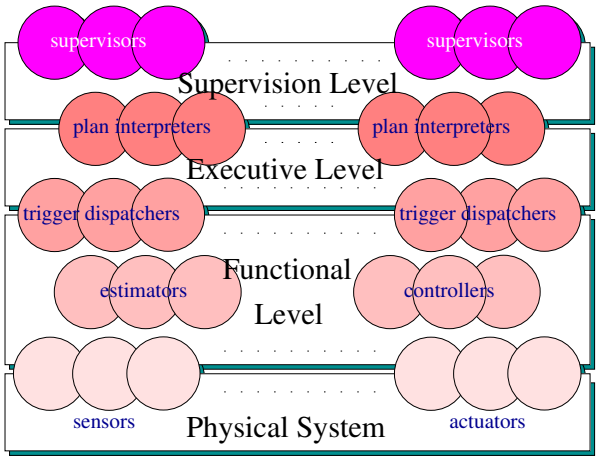
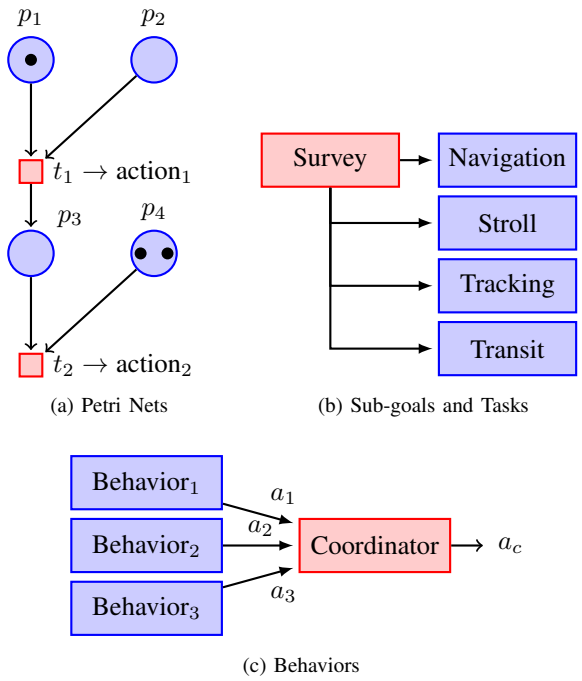


Figure 8: Layered Architecture.

IV. Related Works

MOST research papers describing AUV systems have focused on vehicle's control architectures, and there have been few remarks on how to structure mission specification. However, mission specification architecture is a fundamental aspect, since it constraints the mission spectrum the vehicle can perform and it influences other aspects like robustness, ease of definition, modularity, etc.



(a) Petri Nets (b) Sub-goals and Tasks

(c) Behaviors

Figure 9: Mission specification designs

In Fig. 9 some common mission specification designs are depicted: (a) Petri nets, used in ProCoSa [1] and CORAL [19], [20], [7] systems, (b) tasks, used in ITOCA [21] and others [22], and (c) behaviors [4], [22]. Their main features are summarized below:

1) *Petri Nets:* Petri Net formalism brings robustness and reliability. Its powerful representation allows flexibility and monitoring by means of net marking. Modularity is achieved employing modular Petri nets [7]. Petri net's graphical representation and available building and simulation tools provide portability and ease definition. Furthermore, this representation is suitable for direct interpretation onboard. Their main drawback is the imposed formalism and tools, which need some adaptation to AUV mission domain too.
2) *Tasks:* Sub-goals and tasks mission specification impose DSL development and management, reducing portability. Modularity and monitoring are provided only at task level, while reconfiguration is possible through task configuration parameters.
3) *Behaviors:* Missions specified with behaviors are similar to tasks based ones, but with some important differences. Due to behavior coordination, monitoring becomes difficult, specially if cooperative coordination is needed. It is common to embed learning mechanisms, which in fact allow mission auto-reconfiguration.

Tasks are comparatively easier to specify and monitored than behaviors, so the former is preferable for mission specification. Indeed, many mission specifications are based on tasks [24], [22], [21]. Despite the lack of GUI tools to define missions, translate them into DSL or interpret them, their development time is short. The specification with Petri nets comes with software tools, but they have to be adapted to AUV mission specification. It is also common to find robotic systems that manage behaviors and learning algorithms [4], but their mission specification power is weaker.

Planners are usually embedded in autonomous robotic systems to allow runtime decision making when replanning under uncertainty or failure is demanded. To a large extent, tasks are easily integrated within planners, e.g. in RAP system a task is described by a Reactive Action Package (RAP) which is a context sensitive program specifying a variety of plans for achieving the task in different situations [9]. Most modern AUV architectures as T-REX (Teleo-Reactive EXecutive) goal-oriented system are based on this approach [14], [15]. Embedded automated planning and adaptive execution are T-REX key features, supported by a Constraint-based Temporal Planning approach based on EUROPA$_2$ planning and scheduling solver [12], [2]. Mission specification may support some sort of planner parametrization to allow behavior selection under certain circumstances, by means of action's exception handlers.

We will describe now in more detail some systems that are more closely related to our proposal, as far as they are concerned with mission specification.

A. T-REX (McGaan @ MBARI)

T-REX (Teleo-Reactive EXecutive) [14] is a goal-oriented system, with embedded automated planning and adaptive execution using agents. It encapsulates the operation of a sense-deliberate-act cycle in what is typically considered a hybrid architecture where sensing, planning and execution are interleaved. In order to make embedded planning scalable the system enables the scope of deliberation to be partitioned functionally and temporally inside units called Teleo-Reactors. This discretization tries to guarantee that the current state of the agent is kept consistent and complete during execution. The agent-state is represented as a set of timelines, which capture the evolution of a system state-variable over time in discrete tick units. A timeline is a sequence of tokens that are temporally qualified assertions expressed as a predicate with start and end time bounds defining the temporal scope over which it holds.

Teleo-Reactors in T-REX are characterized by their functional scope, temporal scope and the timing requirements. They are selected for processing agent timelines, according with their functional scope. This process constitutes the basis for inter-reactor communication inside T-REX, using mechanisms such as timeline ownership (internal/external), timeline observation, goal or desired timeline value, and dispatching and notification rules.

A timeline processing algorithm executes as the heart of a T-REX agent at the start of every tick. There are three key steps in the algorithm: 1) all timelines are synchronized at the current execution frontier, 2) new goals are dispatched, and 3) the remaining CPU time can be allocated to reactors for deliberation in incremental steps. Each of these component algorithms operate over the entire set of reactors.

Let's analyze an AUV architecture example proposed by the authors in [15]. There, four Teleo-Reactors are defined to be

responsible for the different control loops operating inside the vehicle: Mission Manager, Navigator, Science Operator and Executive.

The Mission Manager provides high-level directives to satisfy the scientific and operational goals of the mission. Its temporal scope is the whole mission, taking minutes to deliberate if necessary.

The Navigator and Science Operator manage the execution of sub-goals generated by the Mission Manager. The temporal scope for both is in the order of a minute although they differ in their functional scope. Each refines high-level directives into executable commands depending on current system state. The Science Operator is able to provide local directives to the Navigator in case, for example, of detecting something interesting to explore. Deliberation may safely occur at a latency of 1 second for these reactors.

The Executive provides an interface to the underlying AUV functional layer. It encapsulates access to commands and vehicle state variables. The Executive is approximated as having zero latency within the timing model of the application since it will accomplish a goal received with no measurable delay (no deliberation).

B. MOOS (Newman @ MIT)

MOOS [18] refers to a suite of libraries and executables designed and proved to run a field robot in sub-sea and land domains. Included in its scope are a platform-independent communication API, sensor management, state of the art navigation, vehicle dynamic control, concurrent mission task execution, vehicle safety management, mission logging and mission replay.

As far as we know, MOOS is the only programming framework has been used by some groups in programming AUVs. MOOS provides an inter-process communication library. MOOS' processes never communicate directly. Instead, all messages go through a central MOOS server that acts as communication hub, configuring a star-like topology. Every MOOS server together with all processes that communicate through it form a MOOS community and there may be several communities within a system. The communication model proceeds through three actions: *publication*, performed by the process acting as data producer; *subscription*, that must be carried out by the process that will consume some data; and *notification*, issued by the MOOS server to all subscribers whenever a datum is modified.

MOOS provides driver modules to ease the integration of some sensors widely used with AUVs (e.g. GPS, DVL, sonar, altimeter, ...). It also provides modules for navigation, control and data logging, tools for mission replay from log files and communication debugging. It also offers a simple multivehicle simulator.

In MOOS, missions can be defined in terms of tasks. Tasks must belong to certain predefined vehicle's basic capabilities like "GoToWayPoint", "GoToDepth", "LimitDepth" or even to perform some pattern of waypoints. Tasks can be declared and configured statically using a text file. MOOS uses a priority-based scheme to solve arbitration problems among tasks. A mission in MOOS can be specified declaring the sequence of tasks that should be executed and controlled by the mission controller. Missions can be redefined instructing the mission controller to load a different mission file.

Mission plans in MOOS are made up from a set of tasks that synchronize through the exchange of messages. A task can control any number of other tasks sending messages of certain types. In a mission file, tasks are given a unique name, its type is declared and they are configured using the following fields [18]:

- *StartFlags:* A list of messages names that if received will put this task into operation.
- *FinishFlags:* A list of messages that are emitted when the task completes or starves because it is not receiving data.
- *EventFlags:* A list of messages that are emitted when some event happens but the task is not complete.
- *TimeOut:* The maximum time the task should run for. If this timeout expires before the task terminates, Finish-Flags messages are sent and the task terminates. A task can be declared to never timeout.
- *InitialState:* If the task is initially on it does not listen for StartFlags messages. Otherwise, it needs to receive a StartFlags message before it goes active.
- *Priority:* This are used to arbitrate concurrent access to certain resources like actuators. The lower the priority the more important the task is.

Using this combination of message names, priorities and timeout is how a mission is built with MOOS. Functionally, it develops a mission as a network of tasks that are coordinated exchanging typed messages.

V. Summary

In this article we have discussed how a plan-based mission specification may be used to configure a AUV mission. The segmentation of a whole mission into plans simplifies its specification and fosters reutilization. A trigger-based task model for specifying missions abstracts events that allow flexible mission parametrization, modulation, monitoring and control. In comparison with other common mission representations used in AUVs —like Petri nets or behaviors— plan-based missions come up easier to define, as they are closer to AUV mission domain. Moreover, specifying missions using task oriented plans avoids the necessity of onboard planners that might require significant computational costs. In addition, the use of established design patterns for robotic systems fostering separation between the control system and the system under control, have been considered as a key principle in order to ease system architecture design. Furthermore, some of these design patterns have been reinterpreted at design level into a component-based programming framework specifically aimed at developing robotic systems, considering that, in general this kind of frameworks provide already solutions to some of the recurrent problems faced when building robotic control software.

References

[1] Claude Barrouil and Jérôme Lemaire. An integrated navigation system for a long range AUV. *IEEE Oceanic Engineering Society*, (1):1–5, September 1998.

[2] Tania Bedrax-Weiss, Jeremy Frank, Ari Jónsson, and Conor McGann. EUROPA$_2$: Plan Database Services for Planning and Scheduling Applications. November 2004.

[3] Matthew Bennett, Daniel Dvorak, Joseph Hutcherson, Michel Ingham, Robert Rasmussen, and David Wagner. An Architectural Pattern for Goal-Based Control. In *Proceedings of the 2008 IEEE Aerospace Conference*, March 2008.

[4] Marc Carreras Pérez. *A Proposal of a Behavior-based Control Architecture with Reinforcement Learning for an Autonomous Underwater Robot*. PhD thesis, University of Girona, Girona, Spain, May 2003.

[5] Søren Christensen and Laure Petrucci. Modular analysis of Petri Nets. *The Computer Journal*, 43(3), 2000.

[6] Antonio C. Domínguez-Brito, Daniel Hernández-Sosa, José Isern-Gonzalez, and Jorge Cabrera-Gámez. Coolbot: A component model and software infrastructure for robotics. In Davide Brugali, editor, *Software Engineering for Experimental Robotics*, volume 30 of *Springer Tracts in Advanced Robotics*. Springer - Verlag, Berlin/Heidelberg, April 2007.

[7] R. A. Duarte Oliveira. *Supervisão e Controlo da Missão de Veículos Autónomos*. PhD thesis, Universidade Técnica de Lisboa. Instituto Superior Técnico (IST), Lisboa, Portugal, 2003. Dissertação para obtenção do grau de mestre em engenharia electrotécnica e de computadores.

[8] D. Brugali (Editor). *Software Engineering for Experimental Robotics*. Springer Tracts in Advanced Robotics, Volume 30/2007. Springer Berlin/Heidelberg, 2007.

[9] R. James Firby, Roger E. Kahn, Peter N. Prokopowicz, and Michael J. Swain. An Architecture for Vision and Action. In *Fourteenth International Joint Conference on Artificial Intelligence*, pages 72–79, 1995.

[10] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1995.

[11] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In *Proceedings of the 11th International Conference on Advanced Robotics (ICAR'03)*, pages 317–323, Coimbra, June 2003.

[12] Ari K. Jónsson, Paul H. Morris, Nicola Muscettola, and Kanna Rajan. Planning in Interplanetary Space: Theory and Practice. Technical report, NASA Ames Research Center, Brekenridge, 2000. AIPS 2000.

[13] Laboratoire d'Analyse et d'Architecture des Systèmes - LAAS (CNRS). LAAS OpenRobots Project. http://softs.laas.fr/openrobots.

[14] Conor McGann, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn, and Rob McEwen. T-REX: A Model-Based Architecture for AUV Control. In *3rd Workshop on Planning and Plan Execution for Real-World Systems 2007*, 2007.

[15] Conor McGann, Frederic Py, Kanna Rajan, Hans Thomas, Richard Henthorn, and Rob McEwen. A Deliberative Architecture for AUV Control. In *International Conference on Robotic and Automation (ICRA) 2008*, 2008.

[16] Tadao Murata. Petri Nets: Properties, Analysis and Applications. In *Proceedings of the IEEE*, volume 77, 1989.

[17] Issa A. D. Nesnas, Anne Wright, Max Bajracharya, Reid Simmons, Tara Estlin, and Won So Kim. Claraty: An architecture for reusable robotic software. In *in SPIE Aerosense Conference*, 2003.

[18] P. Newman. The MOOS project homepage. [Online ; accessed May 8, 2009].

[19] P. J. C. Ramalho Oliveira, A. Pascoal, V. Silva, and C. Silvestre. Design, development and testing of a mission control system for the marius auv. *Department of Electrical Engineering. Institute for Systems and Robotics*, Instituto Superior Técnico (IST)(1):20, 1996.

[20] P. J. C. Ramalho Oliveira, A. Pascoal, V. Silva, and C. Silvestre. The mission control system of the marius auv: System design, implementation and tests at sea. *International Journal of Systems Science*, 29(10):1065–1080, 1998. Special Issue on Underwater Robotics.

[21] P. Ridao, J. Yuh, J. Batlle, and K. Sugihara. On AUV Control Architecture. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, volume 2, pages 855–860, 2005.

[22] G. N. Roberts, R. Sutton, and R. Allen. Guidance and control of underwater vehicles. *Elsevier Science and Technology*, IFAC Proceedings Volumes(1):1–40, 2003.

[23] C. Schlegel. *Navigation and Execution for Mobile Robots in Dynamic Environments: An Integrated Approach*. PhD thesis, University of Ulm, 2004.

[24] Reid Simmons and David Apfelbaum. A Task Description Language for Robot Control. In *Proceedings of Conference on Intelligent Robotics and Systems*, pages 1–7, March 1998.

[25] D. B. Stewart, R. A. Volpe, and P. Khosla. Design of Dynamically Reconfigurable Real-Time Software Using Port-Based Objects. *IEEE Transactions on Software Engineering*, 23(12):759–776, December 1997.

Balanced Multi-Robot Exploration through a Global Optimization Strategy

Ling Wu, Domenec Puig and Miguel Angel Garcia

***Abstract*— This paper reviews the state of the art in coordinated multi-robot exploration and proposes a new exploration objective based on a practical scenery, reducing the difference of waiting time among different regions of a workspace, which has not been still considered in the literature. A new global optimization strategy for coordinated multi-robot exploration based on a proper dispersion of robots in separate regions is presented. This strategy aims at achieving the lowest variance of regional waiting time and the lowest variance of regional exploration percentage. Both features reveal that the proposed strategy performs better than other state of the art approaches.**

***Index Terms*— Multi-robot exploration, multi-robot coordination, waiting time variance, K-Means.**

I. Introduction

The exploration of unknown areas by means of mobile robots is an important issue that has been widely studied along the last years. Its importance lies in the large number of applications that require robotic exploration, such as search and rescue, planetary exploration, or reconnaissance, among others.

Coordination has been considered in the exploration problem when teams of robots are to be deployed. Multiple robots can carry out tasks faster than a single robot. Nonetheless, a multi-robot system is not n-folded productive due to a variety of factors. First of all, space limitations in the environment can force the robots to move together. Second, multiple robots can interfere with each other [1]. Third, without knowing the existence of the other robots or their locations, a robot could explore the same places that other robots have already searched. Thus, coordination is necessary to increase the system gain in a multi-robot system.

This paper describes a new K-Means based (KME) global optimization strategy for coordinated multi-robot exploration that builds on a previous proposal [2]. The main objective consists of reducing the difference of waiting time among different regions of a workspace. The proposed method is compared to three representative exploration algorithms: Yamauchi [3], Burgard et al. [4], [5], and market economy by Zlot et al. [6]. A statistical study presented in this paper reveals the greedy behavior of those proposals, in the sense that they can explore some parts of the environment much later than others. In contrast, the proposed planner ensures a balanced exploration of the environment by forcing the explicit dispersion of robots over it. This behavior is desirable in a variety of applications, such as search and rescue.

This article is organized as follows. Section II reviews the state of the art on multi-robot exploration. Section III proposes a new team objective for exploration applications and the need for a global optimization strategy. Section IV describes the proposed multi-robot coordinated exploration algorithm. Section V presents the experimental setup, the definition of two new features (the regional waiting time variance and the regional exploration percentage variance) and the experimental comparison among the tested approaches in terms of both features. Finally, conclusions and further improvements are given in Section VI.

Ling Wu, Domenec Puig are with the Dep. of Computer Science and Mathematics, University Rovira i Virgili, Av. Dels Paisos Catalans, 26, 43007 Tarragona, Spain. E-mail: {ling.wu,domenec.puig}@urv.cat

Miguel Angel Garcia is with the Department of Informatics Engineering, Autonomous University of Madrid, Francisco Tomas y Valiente 11, 28049 Madrid, Spain, E-mail: miguelangel.garcia@uam.es

This work has been partially supported by the Spanish Ministry of Education and Science (MEC) under project DPI2007-66556-C03-03. Ling Wu is supported by a FPI scholarship from the Spanish MEC.

II. State Of The Art

This section summarizes different well-known coordinated multi-robot exploration methods. The benefits and defects of those methods are also discussed.

The algorithm proposed by Yamauchi [3] is the simplest exploration strategy. Each robot simply looks for a frontier cell that can be reached with the lowest cost. It is a greedy strategy with no coordination among robots.

The algorithm by Burgard et al. [4], [5] aims at reducing the exploration completion time and thus coordinates the robots to explore as much area as possible. In order to achieve this goal, a decision-theoretic approach trades off the *utility* and the *cost* of visiting targets. The cost of a target is the length of the optimal path that a robot takes to visit it, whereas the utility of a target is the area expected to be found when the robot arrives at it. The utility of a target is reduced if it is inside the sensor range of assigned targets of other robots. Therefore, robots will tend to disperse. However, the degree of dispersion is local, that is, they disperse until the overlap of the expected new exploration areas by different robots is zero. This does

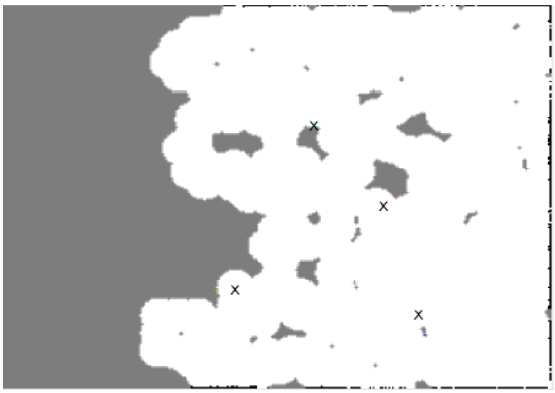


Fig. 1. Example of the greedy behavior of [6]. Four robots (marked with X) start exploring from the bottom right corner (the workspace is a blank area without any obstacles). After quite a number of exploration steps, no robot has arrived at the left part of the workspace. The gray and white areas correspond to unexplored and explored free space respectively.

not guarantee dispersion over the whole workspace. Therefore, robots behave similarly to those of Yamauchi's method [3]. The method does not apply any optimization algorithm to organize the assignment of targets to robots.

The algorithm by Fox et al. [7] deals at the same time with exploration and mapping with multiple robots that do not know their relative positions. The positions of frontiers and hypothetical positions of other robots are considered as targets. The definition of cost and utility for frontiers is similar to [4], [5]. It organizes the robot-target assignment by an optimization scheme, which maximizes the total benefit in each decision step. However, it does not force the dispersion of robots. Therefore, robots can go to different targets that are close to each other. The behavior of robots is similar to [3] as well.

Simmons et al. [8] outperformed [4], [5] in two ways. Firstly, they improved the estimation of the *information gain* that robots are supposed to attain when they arrive at their targets. The *information gain* is the amount of area that a robot is expected to discover at the position of a target, subtracting the overlapped areas expected to be found by other robots. Therefore, it forces robots to disperse as [4], [5]. Secondly, this method proposes the concept of bids. Bids are submitted by individual robots, which calculate them as the expected information gain minus the cost of visiting the goals, similar to the trade-off (benefit) value of [4], [5]. A center agent collects the bids from all robots and assigns the target to the robot which offers the highest bid. This method does not optimize the overall assignment performance.

The approach by Zlot et al. [6], a well-known market-economy based method, is a further improvement to [8]. In [8], robots choose targets from frontiers during the exploration process. Those frontiers lie at the border of the free known space. Instead, in [6], each robot selects some targets from the unexplored workspace so that they form a representative subset of the whole set of future unexplored goals. Targets are auctioned by robots (auctioneers) to others robots (bidders) with proposed prices. Bidders calculate their bids as the expected benefit, which is the expected *revenue* (utility) minus the expected cost. If the bid calculated by a bidder is lower than the proposed price, the bid will not be submitted. If the bids sent by bidders are higher than the proposed price, the auctioneer will choose the bidder with the highest bid. If no bid is submitted for the auctioned job, the auctioneer will keep the job for itself. In short, who offers the highest price for a target will finally win it. If a robot wins the auction of a target, it adds this target to its route list for visiting it in the future.

The auction algorithm is actually an optimization solution to a classical assignment problem in which there are N persons and N tasks, and each task j has a respective benefit value $b_{i,j}$ if it is assigned to person i, and the objective of the problem is defined as finding out the maximized total benefit of an assignment. Bertsekas [9] showed that his auction algorithm can effectively find an optimal solution that maximizes the total benefit. Therefore, the market economy based algorithm is an optimization assignment solution for assigning robots to tasks in each decision step. It is similar to the proposal in [7] or any other that applies an optimal solution for maximizing the total benefit for N pairs of robot-task assignments.

Although the market economy method has applied the above described optimization of total benefit (*utility* - *cost*), and it has tried to improve the global productivity by considering the future targets, it has three problems. Firstly, the method does not explicitly force the robots to disperse. The optimization is undermined similarly to the case of [7]. Secondly, some robots will obtain more targets since they are closer to them (their costs are lower than those of the robots which are farther away from their goals). Consequently, they are always closer to the farther targets in the same direction since they have arrived before other robots and, therefore, the workload is not balanced among the robots in most of times. Thirdly, targets are auctioned again in future decision steps in order to further improve the decision when robots change their positions. However, targets which previously belong to a robot at earlier steps will probably be acquired by another robot if the latter is currently closer to them. Therefore, this approach will finally behave like the greedy strategy [3], in which robots always choose the closest targets. This will be further explained in section III.

The exploration algorithm based on K-Means clustering proposed in [2] has overcome the greedy characteristic of the previous strategies by sending robots to different regions of the workspace. Unknown cells of an occupancy grid are clustered into as many disjoint regions as available robots by applying the K-Means clustering algorithm, with K being the number of robots. Each robot is then assigned to its closest region according to the Euclidean distance from the robot to the region's centroid. Each robot is assigned the frontier cell with the lowest cost, which is calculated by summing: (a) the length of the shortest path between the robot and the frontier cell, (b) a penalization of the Euclidean distance from the frontier cell to the centroid of the region assigned to that robot, provided the frontier does not belong to that region, (c) a constant penalization in case the frontier cell is within the sensor range of another frontier cell assigned to a different robot (this guarantees the repulsion between robots). All

robots start moving to their assigned frontier cells until the first robot reaches its destination. At that point, the target assignment will be decided again.

However, experiments show that some flaws undermine the overall exploration efficiency of that algorithm. Firstly, the assignment of robots to regions can be improved by an optimization strategy instead of by assigning robots to regions on a sequential basis. The latter has been found to cause a longer total sum of routes traveled by robots. Secondly, the estimation of geometric distances between robots and centroids of regions (used to decide the assignment of robots to regions) is a very coarse estimation of the real distance between robots and regions, and it does not fully exploit the actual path length that may be already known. For example, when a big obstacle has already been found between a robot and a region, the shortest route length from the robot to that region through free space is the actual distance between them. If the distance between robots and regions is more accurately defined, the assignment of robots to regions will be more efficient.

Finally, the penalization of the distance from a frontier to the centroid of a region when a robot does not choose a target frontier beside its own region, may cause the robot strongly favors frontiers beside its own regions. Therefore, it will tend to ignore unexplored cells of the regions assigned to other robots that are on its way to its own region. Moreover, since the process repetitively assigns robots to regions during exploration, a robot usually changes its target according to its assigned new region. When a robot is continuously assigned to different regions and since it favors the frontiers of its own region, it can change its target drastically and move back and forth without exploring new area. Hence, the target selection should be stabilized in order to prevent that undesirable behavior. The algorithm proposed in this paper, which is described in section IV, improves the aforementioned aspects.

III. Exploration Objective

In exploration problems, it is not possible to know the optimal routes for the robots until the map of the workspace is revealed. Hence, an approach that makes an optimal choice in every decision step does not necessarily lead to a global optimal solution. When an optimal solution is intractable (since the problem is NP-hard), it is interesting to achieve a close to optimal solution. In order to do that, it is necessary to apply a global strategy to drive the exploration process according to a certain objective. A hierarchy of decision planners should be considered: a planner to achieve the global exploration objective and the planners to refine every decision step. We propose to apply three levels of optimization in a coordinated multi-robot exploration algorithm: (a) path planning strategy; (b) task assignment optimization strategy at every decision step; (c) global optimization strategy in order to overcome the greedy behavior caused by the second level optimization strategy.

The path planning strategy is the lowest level optimization strategy of the exploration algorithm. It decides the path that a robot will follow to arrive at a chosen target. In order to avoid obstacles when robots are traveling, an obstacle-free path should be planned. Furthermore, even if the main objective of the exploration application is not to minimize the total traveling cost of robots, minimizing that cost is still a basic requirement in order to guide robots to travel more efficiently. For example, the approaches presented in Section II, which aimed at minimizing completion time, have applied some kind of optimal route calculation.

Most of the state of the art strategies mainly focus on how to assign tasks to robots. Some of them (such as [6],[7]) applied a second level optimization algorithm. The second level optimization aims at efficiently assigning the robots to the next targets in every decision step. With the information obtained at each moment, although there are uncertainties about the contents of the workspace, the decision maker will try to do its best to assign the targets to the robots in order to minimize the completion time or the total traveling cost. Combinatorial methods can be used in order to make an optimal solution. If the optimal solution cannot be found in polynomial time, a heuristic solution can be used instead.

The highest level optimization aims at compensating the greedy behavior caused by the second level optimization. Without a global view, any optimization algorithm can end up yielding a local optimum. In [3]-[6], each robot tends to move to a target as fast as possible in order to minimize the completion time. Therefore, they all choose the closest targets except that in [3], [4] where dispersion is considered but only at a local level. Robots in the market economy method [6] show similar greedy behavior —exploring some parts of the workspace much later than others (see Fig. 1).

A. Minimizing the Waiting Time Variance for Each Region of Workspace

Based on the above motivation, we propose a new objective function that can be useful for many exploration applications. It consists of minimizing the variance of waiting time of every region in the working space (MINIWTV). Minimizing the waiting time variance (WTV) is an important problem in scheduling [10]. An analogue of this proposition is to minimize the waiting time variance of services in computer networks. When the clients' WTV of network services is minimized, that is, they get all the services after waiting for similar time slices, clients feel that the quality of service (QoS) of the network services is stable and the performance of the network is predictable. In search and rescue applications, if the different regions of the working space can be explored with similar arrival times (minimizing the waiting time variance of different regions), potential victims in a region will not have to wait much longer than those in other regions.

The algorithm proposed in the present paper (KME) aims to achieve the above objective. First, it partitions the unexplored space into as many areas as available robots by applying the technique first introduced in [2]. This partitioning acts as a global optimization strategy. Once the unknown space is

divided into regions, each region is assigned to a robot by a Linear Programming (LP) algorithm in order to minimize the total path cost, considering the relative location of robots and regions. Section IV.B describes how to define the distance from a robot to a region. Section IV.C explains how to decide the next target a robot should visit. These processes act as a second level optimization strategy. Hence, the KME method is a globally optimized approach that reduces the waiting time in

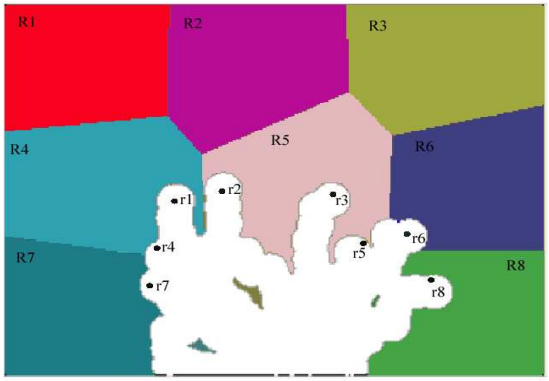


Fig. 2. Example of K-Means clustering algorithm used to partition an empty working space in the proposed multi-robot exploration method. A partition of unknown space into eight regions and robots (K = 8) is shown. Every region is assigned to a single robot. The white area represents space that has already been explored, while the colored regions are the clusters produced by K-Means.

different regions, since robots will disperse to explore different unknown areas partitioned by the K-Means clustering algorithm, avoiding thus the greedy behavior of previous methods, which can partially explore a part of the map long after another.

IV. Coordinated Exploration Algorithm Based on K-Means

The proposed multi-robot coordinated exploration algorithm has several advantageous features comparing to the state of the art approaches. Firstly, robots disperse globally so as to reduce the difference of the waiting times between different regions of the workspace. Secondly, at every moment, the workload distributed to a group of robots is balanced. The applied K-Means clustering divides the exploration tasks into balanced target sets for robots. This favors that different parts of the workspace be explored at a similar speed. Thirdly, exploration is coordinated so as to efficiently reduce the completion time. The optimal assignment of regions to robots is formulated as a Linear Programming (LP) problem and solved by an LP solver [11] so that the total route traveled by the robots is the shortest. This reduces the overall completion time. Furthermore, the dispersion of robots allows them to explore the unknown space in parallel. This also reduces the total route that robots must travel during exploration. When robots are not inside their assigned regions, they will explore the targets that lie on the way to their own regions. When they have arrived at their own regions, the exploration mechanism is similar to that of Burgard et al. [4], [5]. Therefore, it reduces also the completion time of the exploration process. Fourthly, global dispersion and workload balancing are

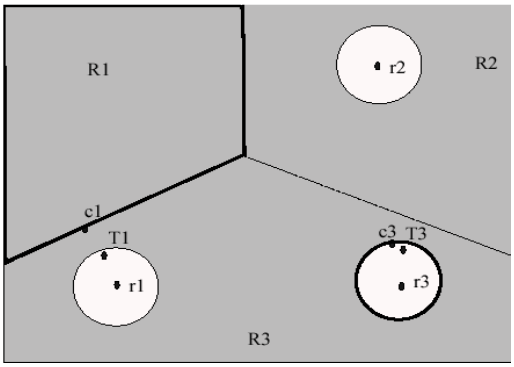


Fig. 3. Gray represents unexplored space that has been partitioned into regions R1, R2, and R3 (three robots r1, r2 and r3 are considered). White represents free spaces that have already been explored by r1, r2 and r3. R2 is accessible for r2. R3 is accessible for r1, r3. R1 is inaccessible for r1, r2 and r3. R1's contour line has been marked with a dark line. For r3, its accessible contour cells of R3, a circle, are marked with a dark line.

dynamically maintained during exploration. This leads to a proper assignment of robots to regions and targets to robots, increasing in this way the overall efficiency of the exploration.

This section is organized as follows. Section IV.A describes the partitioning method. Section IV.B and IV.C present the strategies for optimizing the robot-region and robot-target assignments. The overall exploration algorithm is summarized in Section IV.D.

A. Partition of the Unknown Space through K-Means

In the proposed algorithm, the unknown workspace is partitioned into as many regions as robots by means of the same K-Means based technique first proposed in [2] (see Fig.2). In particular, all unknown cells in the map's occupancy grid are clustered by applying the well-known K-Means algorithm. The iterative process is described in [2], [12]. Since the unexplored space is partitioned into regions and they will be explored by different robots in parallel, the workload is distributed among robots in a balanced way and the waiting time of each region in the working space is reduced. Thus, for search and rescue applications, victims dispersed over the working space will be found by robots that are in their vicinities more quickly. By working simultaneously in separate parts of a map, robots keep a reasonable dispersion among them. In other solutions that do not apply such a global strategy (e.g., [3]-[6]), the robots may well concentrate in some parts of the workspace, reaching other parts significantly later (see Fig.1).

B. Robot-Region Assignment

After the unknown space is divided into K regions, the algorithm decides which region is assigned to each robot (see Fig.2 and Fig.3). The proposed algorithm distinguishes between two types of regions and defines the distance for each case. In the first case, when a region is *accessible* (*RA*) for the robot, it can move straight ahead to a free cell which lies at the verge of this region. The route is calculated by a method like the one proposed in [4], [5]. This algorithm will automatically avoid collision with walls and have the shortest cost, which will be referred to as real path cost. The distance from a robot

to this region is defined according to this cost.

In the second case, if there is no free space lying between a robot and a region, the robot will first explore other unknown regions before reaching its assigned region, therefore, the region is *inaccessible* (*RI*) for it. In Fig. 2, for example, the upper three regions are inaccessible for all robots. The distance from a robot to an inaccessible region is a geometric distance estimated as described next.

First of all we define the distance between a robot *r* and a contour cell *c* of a region *R*. A contour line is the set of unexplored cells that lie on the edge of *R*. A frontier cell *f* is an explored free cell that lies beside a contour line. Robots can go to a frontier cell and use their sensors to explore a new area. If a contour cell *c* has free neighboring frontier cells F_c such that *r* can go directly to *c* through free space, then *c* is accessible for *r*. If *R* has accessible contour cells for *r*, *R* is accessible for *r* (note that even if a contour cell is beside free space, but *r* is not in that space, *R* is still not accessible for *r*). The distance between *r* and an accessible contour cell c_a of *RA* is defined as $d(r, c_a)$, the shortest distance from *r* to all possible frontier cells adjacent to *c*.

$$d(r,c_a) = \min\{\delta(r,f) \,|\, f \in F_c\} \qquad c_a \in RA \tag{1}$$

where $\delta(r,f)$ is the real path cost from *r* to a cell within F_c, when *c* is accessible for *r*.

When a contour cell is in an inaccessible region *R* for *r* (it is denoted as c_i), since *r* can not arrive at it immediately, an estimation of distance is defined as the geometric distance between *r* and *c*. If there is an obstacle on the direct line connecting *r* and c_i, the distance is penalized with the diagonal length of the map, *l*. The distance from *r* to a contour cell c_i of *RI* is defined as:

$$d(r,c_i) = g(r,c) + \rho \qquad c_i \in RI \tag{2}$$

The penalizing value ρ will give priority to the contour cells that do not have an obstacle blocking the direct line between the robot and them. Thus, ρ is 0 if there is no obstacle between *r* and *c*, otherwise, it is equal to *l*.

Finally, the distance between *r* and *R* is calculated as:

$$d(r,R) = \min_c\{d(r,c)\} \qquad c \equiv c_a \quad or \quad c \equiv c_i \tag{3}$$

Having defined the distance from a robot to a region, it is possible to determine the closest region to a robot. In some cases, two or more robots are closer to the same region than to other regions. If there were no control on the assignment of regions, all these robots could end up exploring the same region. Therefore, it is necessary to design an optimal solution to assign robots to regions fairly. In this proposal, the assignment between robots and regions is formulated as a Linear Programming (LP) problem. An LP solver [11] is applied to obtain an optimal solution.

If there are *N* robots, the optimization objective consists of finding out an *N*-tuple of robot and region pairs $\left((r_1, R_1'), (r_2, R_2'), \ldots (r_n, R_n')\right)$ such that the total sum of robot to region distances is minimized. R'_i represents the region that is assigned to r_i.

Let $a_{ij} = 1$ denote that region R_j is assigned to r_i, and $a_{ij} = 0$ that R_j is not assigned to r_i ($i,j = 1,\ldots,N$). The problem is defined as finding a set of a_j ($i,j = 1,\ldots,N$) subject to:

$$\sum_{i=1}^{N} a_{ij} = 1, \qquad \sum_{j=1}^{N} a_{ij} = 1 \qquad i, j = 1,.., N \tag{4}$$

that minimize the sum of distances between robots and regions:

$$\sum_{j=1}^{N}\sum_{i=1}^{N} a_{ij}\, d(r_i, R_j) \tag{5}$$

This optimal assignment ensures that the total sum of path lengths of robots moving to their assigned regions is minimized. In the case of Fig.3, r_1, r_2 and r_3 are assigned to R_1, R_2 and R_3. The assignment of regions to robots is adjusted during the exploration so as to minimize the costs of driving the robots in order to explore their corresponding regions. This optimization algorithm is iteratively applied since the relative positions between robots and regions keep changing as new areas of the workspace are being discovered.

C. Robot-Target Pair Assignment

Having decided the assignment of a robot to every region, this section describes to which frontier cell each robot should travel. If the robot's region is accessible for it, it will go to a frontier cell that is beside the closest contour cell of its own region. If a robot's region is not accessible for it, since it will have to travel through the regions of other robots, it is necessary to decide which frontier the robot should visit first.

To reduce the large number of frontiers in the exploration map, a set of *M* frontiers is selected, *FM*, such that each frontier in *FM* stays away from each other at least the sensor range. Each robot should then choose a frontier from *FM* different from the frontiers chosen by the other robots.

Let R_{r_i} be the optimal region assigned to robot r_i according to the scheme described in Section IV.B, and let cc_{r_i} be the contour cell of R_{r_i} closest to r_i. In the case of Fig. 3, c_1 is cc_{r_1} of R_1 for r_1, c_3 is cc_{r_3} of R_3 for r_3.

The goal of the current assignment is to evaluate all the candidate frontiers from *FM* for every robot r_i and find the best target for it. Denote each candidate frontier as f_j ($j = 1, 2,\ldots,M$), and the target frontier for r_i as T_i.

Since the robots whose regions are inaccessible for them need to reach their regions as soon as possible, they have priority to choose their targets first. For those whose regions are inaccessible, equation (6) evaluates the distance between a robot r_i and each frontier of *FM*, $d(r_i, f_j)$. Its value is determined if f_j is accessible for r_i from the robot's position. Otherwise, it is infinite.

$$d(r_i,f_j) = \begin{cases} \delta(r_i,f_j) + \rho(f_j, cc_{r_i}) & if\ f_j\ is\ accessible\ for\ r_i \\ \infty & otherwise \end{cases} \tag{6}$$

where $\rho(f_j, cc_{r_i}) = g(f_j, cc_{r_i}) + \rho_1(f_j, cc_{r_i}) + \rho_2(f_j)$

In (6), δ denotes the real path cost. Since it is possible that the space between a candidate frontier f_j and a robot's closest contour cell cc_{r_i} is unexplored (in Fig.3, T_1 is a frontier cell

and between T_l and c_l there is unexplored space), the geometric distance g is used. Functions ρ_1 and ρ_2 add penalty values to the distance $d(r_i, f_j)$ when f_j is not favorable for r_i.

If there is an obstacle along the line from f_j to cc_{r_i}, $\rho_1 = l$, otherwise $\rho_1 = 0$. If f_j has been chosen as a target by another robot, $\rho_2 = v$, otherwise $\rho_2 = 0$.

Constant v is a positive value. Adding it will force robots not to choose the same frontiers as the ones which have already been chosen by other robots. Constant l is the same diagonal length mentioned in Section IV.B.

The penalty values distinguish the frontiers from each other. If there are two frontiers (f_1, f_2) which are close to each other, and there are obstacles lying on the lines from a robot's closest contour cell cc_{r_i} to f_1 and f_2, they will both get a penalty value l, such that the one with the smallest distance will win. If there is another f_3 which does not have an obstacle lying on the line from it to cc_{r_i}, it will win both f_1 and f_2.

The above evaluation process starts with the first robot r_i (robots are ordered from r_1, to r_N) whose own region is inaccessible. For each robot, the frontier whose distance to the robot is the smallest will be chosen as the target T_i for it. For instance, in Fig.3, T_1 and T_3 are assigned to r_1 and r_3 respectively.

$$T_i = \arg\min_j (d(r_i, f_j)) \tag{7}$$

In the next step, equations (6)-(7) are used again for finding the target frontier for robots whose regions are accessible. There is no much difference between the selection process for robots whose regions are accessible and inaccessible, since $g(f_j, cc_{r_i})$ will make a robot tend to choose the frontier which belongs to its own region and close to the closest contour cells of its own region. Therefore, any robot r_i whose own region is accessible will probably go to the frontier beside cc_{r_i}. Only in the case that there is a frontier lying on the way to cc_{r_i}, the robot will visit it.

D. Exploration Algorithm

The proposed exploration approach is described below:

1: Initialize exploration map, real path map, and K-means label map.
2: **While** there are unexplored cells **do**
3: **If** iteration step is lower than 2, **then** initialize centroids of K-Means randomly from the workspace
4: Search for frontiers beside unknown regions, extract a set of *FM* from the frontiers.
5: Calculate the real path distance from robots to all free cells and store them in robots' real path map
6: Call K-Means to partition the unknown cells and frontier cells and label these cells by the region identifiers in K-means label map
7: Calculate one distance between robots and regions
8: Call LP-solver to assign robots to regions
9: For robots whose regions are inaccessible, calculate the distance from each robot to each frontier, find the smallest distance and assign the correspondent frontier to the robot
10: For robots whose regions are accessible, calculate the distance from each robot to each frontier, find the lowest distance and assign the correspondent frontier to the robot
11: Robots move to their targets until one robot arrives at its target
12: Robots sensor the environment and update the exploration map
13: **end while**

V. EXPERIMENTS

The multi-robot exploration algorithm proposed in this paper and three alternative approaches representative of the state-of-the-art (Yamauchi [3], Burgard et al. [4], [5], Zlot et al. [6]) have been extensively tested in simulation. Sections V.A to V.C describe the experimental setup, the concepts of regional waiting time and waiting time variance, and the experimental results.

A. Experimental Setup

The types of workspace on which the experiments have been carried out are illustrated in Fig.4. The four algorithms are run on all three maps with two starting schemes for the robots: (a) all robots start from the same location, (b) robots start from random locations (see Fig. 5(left)(middle)). Two to eight robots have been tested in order to find out the influence of the number of robots on the performance of the algorithms when the workspace is divided into different number of regions. For every combination of robots, maps and starting schemes, 15 set-ups of runs have been performed. When robots start together, they do it from one of the 15 locations shown in Fig. 5(right). When robots start with the separate scheme, the robots' initial positions are randomly chosen from those 15 locations.

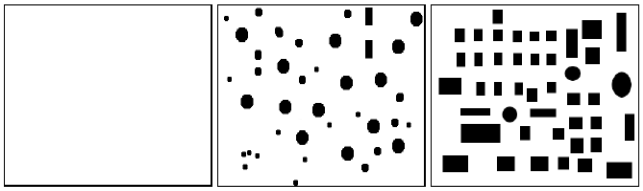


Fig. 4. Three types of workspace (maps): (left) blank map (middle) unstructured map (right) cityblock map.

An experimental configuration includes the combination of a number of robots between 2 and 8, a map out of three, a starting scheme out of two and a starting location out of 15. Hence, the four tested exploration algorithms have been compared over 630 different configurations (3 maps x 2 starting schemes x 7 applied numbers of robots x 15 runs).

The objective of the experiments is to find out how the four exploration approaches behave with respect to the criteria of waiting time variance and exploration percentage variance. The two criteria are defined in the next subsection.

B. Waiting Time Variance and Exploration Percentage Variance

When a robot arrives at a cell at step t and explores its surrounding area with its sensor, suppose cell p is explored, no matter if it is found free or occupied. The *waiting time stamp* (w) of p is set to t. At the end of the exploration, each cell is

marked with a definite *w*. At each step *t*, some cells have already been explored (known cells), whereas the other cells have not (unknown cells). *w* is recorded for every known cell. For every unknown cell, the *estimated waiting time stamp* (*w′*) is calculated as ($w' = e + t$), where *e* is the number of estimated steps from a closest robot to the unknown cell (the distance divided by the cell size).

Let *R* be a region of the workspace that contains *n* known

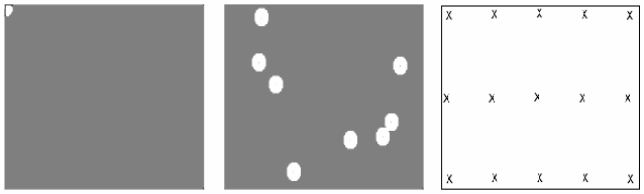


Fig. 5. Starting schemes: (left) robots start together, (middle) robots start from random separate places. Gray represents cells that are unknown for robots. White represents cells that have been explored by the robots' sensors and have been found to belong to the free space. (right) 15 set-ups of the robots' starting positions.

cells and *m* unknown cells. The regional waiting time w_R of that region at time *t* is defined as:

$$w_R = (\sum_{i=1}^{n} w_i + \sum_{j=1}^{m} w'_j) / (m + n) \quad (8)$$

Without loss of generality, the maps are divided into 2, 4, 8, 16, 32, 64 equal-sized regions and the program calculates w_R for each region. The variance of regional waiting time of *N* regions is simply defined as:

$$WTV = \sigma^2(W_{R_i}), \quad i=1,...,N \quad (9)$$

The lower the WTV is, the more balanced the waiting time of regions is. When the exploration process terminates, every cell of the workspace is known and has a waiting time *w* value. The final average waiting time for a region *R* with *n* cells becomes: $awt_R = (1/n)\sum_{i=1}^{n} w_i$

The final WTV becomes the variance of average waiting times of all regions:

$$WTV_{final} = \sigma^2(awt_{R_i}), \; i=1,...,N \quad (10)$$

The exploration percentage of a region (*EP*) at each time step is simply how much percentage has been explored in that region. The variance of *EP* of all regions (*EPV*) gives an explicit measure of the concurrency in exploring the different parts of the workspace.

C. Experimental Results

Fig. 6 shows the mean WTV of eight regions for 15 runs on the unstructured map and the together scheme considering the four tested exploration approaches. YMC, Burgard, Zlot and KME represent Yamauchi [3], Burgard et al. [4], [5], Zlot et al. [6] and the proposed algorithm respectively. The four approaches have been compared under the same configurations. The WTV curves show that the more robots, the lower the WTVs of all approaches. When more than two robots are utilized, KME has the lowest WTV during the exploration, meaning that it explores the different regions with more similar regional waiting times. In addition, WTV_{final} of

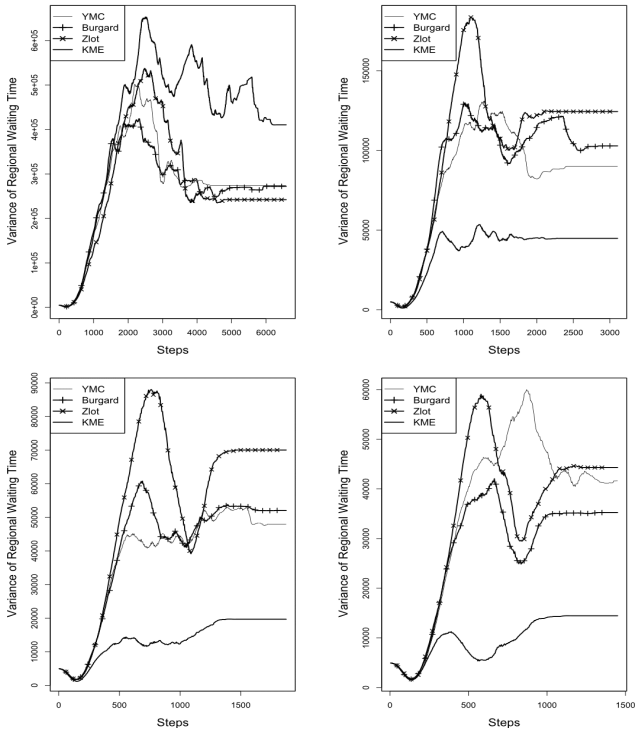


Fig. 6. Curves (mean of 15 runs) of WTV of eight regions and exploration scheme "unstructured map / robots start together". Subfigures from left to right respectively show the curves for 2, 4, 6, and 8 robots.

regions (the value where each curve terminates) with KME is also the smallest except for the two-robot case, which means that the *awt* of different regions are the most similar with KME.

The figures for the same experiments with the workspaces being divided into two, four, sixteen, thirty-two and sixty-four regions are not shown in the paper for space limitations and can be found in the following link (http://deim.urv.cat/~rivi/KMEfigures.html)

From the comparison of WTVs for different divisions of regions, it can be noticed that with the same number of robots (2, 4, 6 and 8 robots), as long as the number of regions is not much bigger than the number of robots, KME always leads to the lowest waiting time variance of all the regions. Hence, the waiting time of all regions is more equalized than with the other three approaches. The same trend can be found in the six combinations of maps and starting schemes.

Fig. 7 shows the results for the unstructured map workspace, and 2, 4, 6, and 8 robots starting from separate locations. The WTV curves of all techniques with the separate scheme are lower than the ones with the together scheme. This is reasonable as when robots are initially separate, they naturally disperse. Thus, the regional waiting time among regions is more equalized. With the separate scheme, KME is again superior to the other methods.

In the figures discussed above, each curve represents the average of the 15 runs. In order to show that a curve obtained by averaging the WTV curves of 15 runs is general enough to observe the general trend, 100 experiments have been carried out under the same conditions. In each of the 100 runs, the

robots start from a random location. The mean of WTV curves for 1, 5, 10, 15, 20, 30, 50 and 100 runs with the workspace divided into 16 regions have been drawn (see the aforementioned link). They show the same trend as the above figures. KME still has the lowest WTV in those experiments. The curves become smoother when the number of runs increases. The only exception is the Zlot's method, which

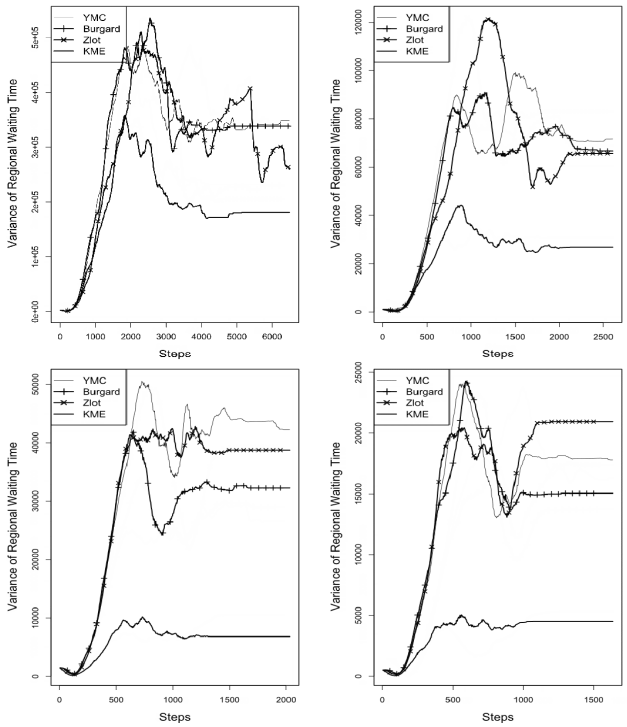


Fig. 7. Curves (mean of 15 runs) of WTV of eight regions and exploration scheme "unstructured map / robots start separately". Subfigures from left to right respectively show the curves for 2, 4, 6, and 8 robots.

grows beyond the Yamauchi's method when more than 30 runs are averaged.

Fig. 8 shows the experiments with respect to EPV with 2, 4, 6, and 8 robots exploring the unstructured map with the robots starting from different locations. The four approaches have been compared with respect to the exploration percentage of eight regions under the same conditions (same map, starting scheme and starting locations). Each subfigure illustrates the mean of EPV of 15 runs during the exploration. Notice that with 2, 4, 6, and 8 robots, the eight regions are explored at a similar speed with KME, whereas even with more than 2 robots, the EPV curves of the other three methods show a big difference of exploration percentage between two regions (see more results in http://deim.urv.cat/~rivi/KMEfigures.html).

Notice that no matter the number of regions, KME is the approach that explores the different regions with the most equalized speed. When the workspace is divided into 2^n regions, if the number of robots is lower than n, KME gradually behaves similarly to the other methods with respect to EPV. Hence, when the number of robots is bigger than n and the number of regions is not too high compared to the number of robots, KME still performs better than the other methods.

On the other hand, Zlot's algorithm behaves like [3]-[5]

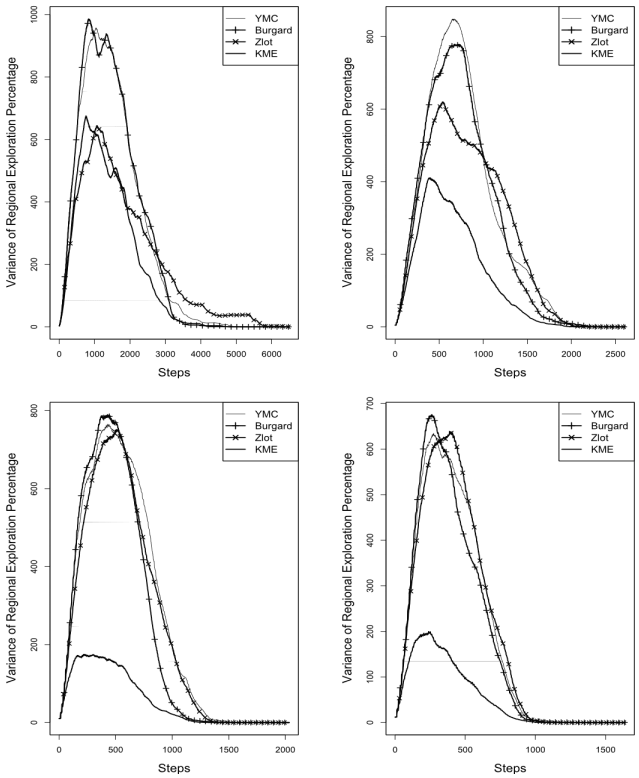


Fig. 8: Curves (mean of 15 runs) of EPV for eight regions and exploration scheme "unstructured map / robots start separately". Subfigures from left to right respectively show the curves for 2, 4, 6, and 8 robots.

with respect to the WTV and EPV. It is clear to see from the descriptions of the methods [3]-[5] that the robots do not disperse or disperse in just a local level. The results presented here show that the Zlot's algorithm does not force robots to disperse into the whole workspace more than the other algorithms [3]-[5]. Although it applies an optimization method for task assignment, its solution is not superior to [3]-[5] with respect to the waiting time variance.

VI. Conclusions

A new algorithm (KME) for multi-robot exploration has been presented in this paper. The proposed algorithm leads to the lowest variance of average waiting time of regions when compared to the state of the art approaches tested in this work, that is, different regions are explored with the most similar average waiting time, this being an important benefit when an application requires a balanced exploration of different regions of the workspace with a number of robots inferior to that number of regions. This behavior is desirable for many exploration applications, such as search and rescue.

References

[1] M. Mataric, "Minimizing complexity in controlling a mobile robot population", *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 1, 1992, pp. 830-835.

[2] A. Solanas, M. A. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space", *Proc. Int. Conf. on Intelligent Robots and Systems*, vol.1, 2004, pp. 717-721.

[3] B. Yamauchi, "Frontier-based exploration using multiple robots", *Proc. of Int. Conf. on Autonomous Agents*, 1998, pp. 47-53.

[4] W. Burgard et al., "Collaborative multi-robot exploration", *Proc. Int. Conf. on Robotics and Automat.*, vol.1, 2000, pp. 476-481.
[5] W. Burgard, M. Moors, C. Stachniss, F. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, 2005, pp. 376-386.
[6] R. M. Zlot, A. T. Stentz, M. B. Dias, S. Thayer, "Multi-robot exploration controlled by a market economy", *Proc. Int. Conf. on Robotics and Automation*, vol. 3, 2002, pp. 3016-3023.
[7] D. Fox et al., "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, 2006, pp. 1325-1339.
[8] R. Simmons et al., "Coordination for multi-robot exploration and mapping", *Proc. Conf. on Artific. Intel.*, 2000, pp. 852-858
[9] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems: a tutorial," *INTERFACES*, vol. 20, no. 4, 1990, pp. 133-149.
[10] N. Ye, X. Li, T. Farley, X. Xu, "Job scheduling methods for reducing waiting time variance," *Computers and Operations Research*, vol. 34, no. 10, 2007, pp. 3069-3083.
[11] Lp-solve, Available: http://lpsolve.sourceforge.net/5.5/
[12] L. Wu, M.A. Garcia, D. Puig, and A. Sole, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, vol. 1, no1, July 2007, pp. 37-44.

Affine image region detection and description

R. Vázquez-Martín, R. Marfil and A. Bandera

***Abstract*—This paper describes a novel approach for affine invariant region detection and description. At the detection stage, a hierarchical clustering mechanism is employed to group image pixels into regions. This process is based on the Bounded Irregular Pyramid (BIP) and takes into account a colour contrast measure, internal region descriptors and attributes of their shared boundaries. High-contrasted regions are selected as salient regions. On the other hand, geometrically and photometrically normalized regions are represented by a kernel-based descriptor. The lenght descriptor is reduced by applying Principal Component Analysis (PCA). The protocol proposed by Mikolajczyk et al. [17], [18] has been conducted to compare the proposed approach with other similar methods. Experimental results prove that the performance of our proposal is high in terms of computational consuming and distinguished region detection and description abilities.**

***Index Terms*—salient regions, feature detection, affine invariant regions, feature description.**

I. Introduction

IMAGE matching is defined in artificial vision as the process of bringing two images into agreement so that corresponding items in the two images correspond to the same real, physical region of the scene. The similarity may be applied to global features derived from the original images. However, this is not the most robust solution when images are taken from different viewpoints. In this work, the image matching problem is accomplished from a feature-based strategy, where images are analyzed first in order to extract some distinguished features. Detected features or image regions are then characterized by a descriptor which will be subsequently employed to solve the matching problem.

In this paper, we propose a novel approach for affine, distinguished image regions detection and description. The core of the detector is a hierarchical algorithm for perceptual grouping of the image pixels. Image segmentation is not a robust process, and obtained results can change depending on the illumination conditions or the viewpoint. However, there are a set of image regions whose properties allow them to be robustly detected in despite of these changes. The aim of the proposed approach is to find these salient regions. On the other hand, to solve the correspondence problem stated when different views of the same scene are compared, detected image regions can be characterized using information obtained from the image. In our proposal, a weighted histogram is employed. This descriptor provides satisfactory results, specially when used in real acquired images. However, to reduce the lenght of the obtained descriptor, PCA is applied.

R. Vázquez-Martín, R. Marfil and A. Bandera are with Department of Electronic Technology, University of Málaga, Campus de Teatinos 29071-Málaga, Spain.
E-mail: rvmartin@uma.es

A. Related work

Given a set of images taken from different viewpoints, the process of finding the projections on each image of real 3D surface patches can be useful for a large number of applications, such as object recognition, robot localization or wide baseline matching for stereo pairs. Among other issues, this process must deal with the problem that image regions associated to the projections change covariantly with the class of transformation induced by the viewpoint change. When the viewpoint change can be approximated by an affine transformation, approaches which solve this problem are called affine region detectors [18].

The detection of regions which change covariantly with affine transformations was described in detail by Mikolajczyk et al. [18]. In this work, the authors provide a review of affine covariant region detectors, and compare their performance on a set of test images under varying imaging conditions. The requirement for these detectors is that they must provide regions whose shapes depend on the underlying image features, so that they correspond to the projections of the same 3D surface patch on the different images. Although the boundaries of these covariant regions do not have to be associated to changes in image features such as colour or texture, some of the approaches described in [18] look for these abrupt changes. Thus, the intensity extrema-based region detector (IBR) [19] starts from intensity extrema and studies a intensity-based function along rays emanating from this extrema to define a region of arbitrary shape. The region is delineated by the image points defined over these rays where the intensity suddenly increases or decreases. A *maximally stable extremal region* (MSER) [15] is a connected component of an appropriately thresholded image where all internal pixels have either higher or lower intensity than all the pixels on its outer boundary. Among these extremal regions, the *'maximally stable'* ones are those corresponding to thresholds were the relative area change as a function of relative change of threshold is at a local minimum.

To match the projections of a real 3D surface on a set of images taken from different viewpoints does not only require to find distinguished image features, but also to solve the correspondence problem established among these sets of features. This issue can be addressed by characterizing the distinctive regions in terms of certain patterns. For this reason, the computation of feature descriptors is done as a separate step from that of feature detection. For local interest point (corner) detection, features are usually described using their associated image patches. Then, Normalized Sum-of-Squared-Differences (NSSD) is employed to find the best matchings. On the other hand, descriptor for scale invariant features are computed at the distinctive points with the associated

scale. Gaussian derivatives computed at the characteristic scale over image patches can be employed to achieve invariance to image rotation. However, among the large number of proposed techniques, the distribution-based descriptors are probably the most used ones. Thus, inside the scale invariant feature transform (SIFT), Lowe [11] proposed to compute a histogram of local oriented gradients around the interest point and scores the bins in a 128-dimensional vector. Mikolajczyk and Schmid [17] proposed a variant of SIFT, called gradient location and orientation histogram (GLOH), which has proved to be more distinctive but also computationally more expensive. The SURF approach includes a region descriptor which uses a distribution of Haar-wavelet responses within the interest points neighbourhood [1].

B. Overview of the proposal

This paper describes a novel approach for affine region detection which extends the idea of looking for abrupt changes. However, instead of changes in intensity or colour (edges), our approach looks for image boundaries which delimitate high-contrasted regions of data-dependent shape. To detect these boundaries, we use a hierarchical clustering scheme which presents two stages: firstly, it groups neighbour image pixels into blobs of homogeneous colour and then, it merges these blobs using a more complex similarity criterion. Basically, this criterion complements a contrast measure defined between regions with image edges detected using the Canny detector, with internal region descriptors and with attributes of their shared boundaries. Finally, it must be noted that the hierarchical clustering algorithm represents the input image at different levels with decreasing resolution. This hierarchy constitutes a scale-space representation where salient regions could be detected at different scales. On the other hand, to describe the detected regions, we have chosen as feature space the colour probability density function (pdf), which must be estimated from the region data. To reduce the computational cost, n-bin histograms are employed. Besides, in order to take into account the spatial information and not only the spectral one, geometrically and photometrically normalized salient regions are characterized by spatially masking them with an isotropic kernel. Finally, we have applied PCA to the obtained kernel-based histograms to reduce its large lenght.

C. Summary and comparison with other approaches

Unlike affine region detectors based on interest point detectors such as the Harris–Affine or the Hessian–Affine techniques [18], our proposal provides complementary image information. Thus, it is more closely related to those region detectors based on image intensity analysis, such as the MSER and IBR approaches. The main difference with those approaches is that it searches for high contrasted regions of uniform properties using a hierarchical structure instead of intensity extrema in a 2D image as do the MSER and IBR detectors. Using this segmentation strategy, it is possible to work in scale-space, improving repeatability for significant scale changes.

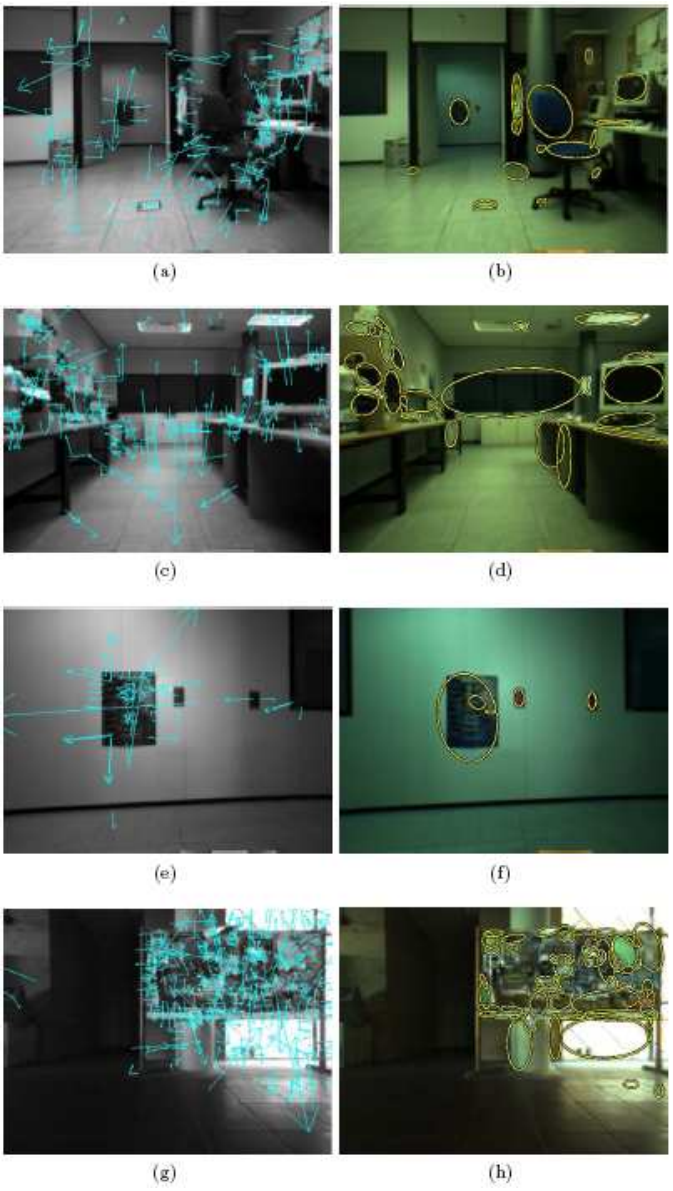


Fig. 1. a-c-e-g) SIFT features and b-d-f-h) affine covariant regions detected by the proposed approach in different environments and situations.

On the other hand, Fig. 1 illustrates the main difference with respect to the popular Difference–of–Gaussians (DoG) detector, the scale invariant feature detector used by SIFT. Typically, the DoG provides an immense number of keypoints, as can be seen in Figs. 1a-c-e-g. On the contrary, the proposed approach detects a far smaller set of regions. This is due to the grouping process inherent to any region–based feature detector. In this case, this grouping merges different image blobs in accordance with their similarity in colour and the shared boundary. These regions have a greater underlying semantic significance than the keypoints detected using a scale invariant detector. Using the proposed approach the image is described by a more organized set of features that allows a reliable matching since comparatively little information is needed to describe an scene.

The rest of the paper is organized as follows: The proposed

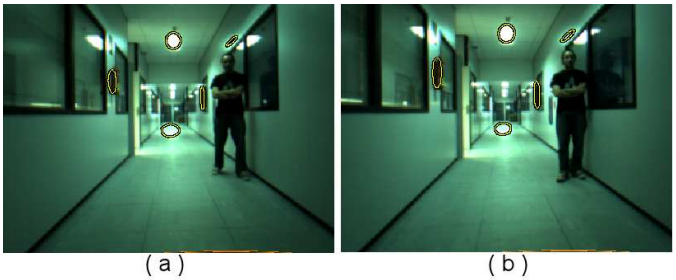


Fig. 2. a-b) Regions generated by the proposed detector on two images taken from a mobile robot. Representing ellipses have been chosen to have the same first and second moments as the originally arbitrarily shaped region (see Fig. 11 for more examples).

approach for the acquisition and description of salient regions is described in Sections II and III. Experimental results are provided in Section IV, where the segmentation algorithm is evaluated and it is also provided an example of the application of the approach in an environment mapping framework. The results of a comparative study of the proposed approach with other similar approaches are given in Section V. Finally, the paper concludes along with discussions in Section VI.

II. Hierarchical clustering approach for salient regions detection

The proposed approach employs a hierarchical graph-based clustering algorithm to detect the high-contrasted regions of the input image. In this hierarchy, the input image defines the base level, which is arranged as a graph where each pixel is a node and neighbourhood relationships are encoded as arcs (intra-level arcs). Upper hierarchy levels are encoded as undirected graphs where the nodes are generated by grouping a set of nodes of the level below and the arcs encode their adjacency relationships. If intra-level arcs represent the neighbourhood of each node at the same level, another set of arcs establish a dependence relationship between each node of level l+1 and a set of nodes at level l. These relationships may be extended by transitivity down to the base level. The set of pixels linked to a node is named its receptive field. The receptive field defines the embedding of this node on the original image. This hierarchy defines an irregular pyramid [5], [8], where each level l is a graph $G_l = (N_l, E_l)$ consisting of a set of nodes, N_l, linked by a set of intra-level edges E_l. In order to speed up the hierarchical clustering process, the employed irregular pyramid combines the classical irregular simple graph with a regular structure. This regular decimation process is only applied in the homogeneous parts of the image. Then, each graph G_l has a regular part which built from G_{l-1} using a 2x2/4 regular decimation procedure and an irregular part which is built from G_{l-1} using an union-find decimation process. This also implies that there are two types of nodes in our structure: nodes belonging to the 2x2/4 regular part (regular nodes) and nodes belonging to the irregular part (irregular nodes). Experimental results demonstrate that the shape of the obtained salient regions is adapted to real items of the scene, being no affected by the regular tessellation (see Fig. 2).

Each level of the proposed pyramid is computed in three steps:

- 2x2/4 regular decimation process: if four regular adjacent nodes of level l have similar colour, a new regular node is created at l+1.
- Irregular node generation process: any regular or irregular node of level l which is not linked to a node at l+1 is included in a union-find grouping process [15], [14]. This union-find process only generates irregular nodes at level l+1.
- Intra-level edge generation in G_l+1: the edges of G_l+1 are computed taking into account the neighbourhood of nodes in G_l.

In order to speed up the building process, the pyramid can be initialized with a first image partition. This initial partition divides the input image in a set of homogeneous regions, constituting an over-segmentation of the input image. Typically, this pre-segmentation process only generates the first pyramid level, and the rest of levels are built using a more complex grouping criterion. Our proposal accomplishes the pre-segmentation and the subsequent clustering process into an irregular pyramid as two consecutive stages. The first stage employs a colour distance to group the image pixels into a set of blobs whose spatial distribution is physically representative of the image content. It must be noted that the hierarchy automatically stops growing when it is no longer possible to link together any more nodes because they are not similar in colour. The set of nodes which are not linked to any node at upper levels define a partition of the input image (see Fig. 3). Then, the second stage clusters the set of homogeneous blobs into a smaller set of regions taking into account not only the internal visual coherence of the obtained regions but also the external relationships among them. Two constraints are taken into account for an efficient grouping process: first, although all groupings are tested, only the best groupings are locally retained; and second, all the groupings must be spread on the image so that no part of the image is advantaged. For managing this grouping, the pyramid structure is used: the roots of the pre-segmented blobs are considered as irregular nodes which constitute the first level of the grouping multiresolution output. However, if the distance between two nodes in the pre-segmentation stage is based on a colour criterion, in order to achieve this second grouping process, a more complex distance must be defined. This distance has two main components: the colour contrast between image blobs and the edges of the original image computed using the Canny detector. Then, the distance between two nodes n_i and n_j, $\Upsilon(n_i, n_j)$, is defined as

$$\Upsilon(n_i, n_j) = \frac{d(n_i, n_j) \cdot min(b_i, b_j)}{\alpha \cdot c_{ij} + \beta(b_{ij} - c_{ij})} \quad (1)$$

where $d(n_i, n_j)$ is the colour distance between n_i and n_j, b_i is the perimeter of n_i, b_{ij} is the number of pixels in the common boundary between n_i and n_j and c_{ij} is the set of pixels in the common boundary which corresponds to pixels of the boundary detected by the Canny detector. α and β are two constant values used to control the influence of the Canny edges in the grouping process. We set these parameters to 0.1 and 1.0 respectively.

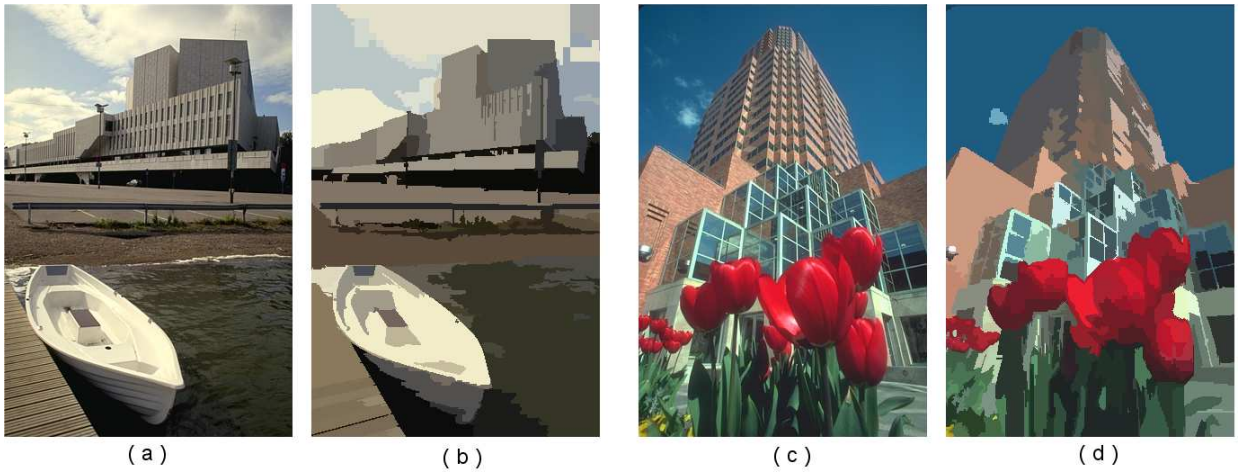


Fig. 3. Presegmentation stage: a-c) Original images; and b-d) colorized pre-segmentation images.

When the whole irregular pyramid is built, there is a set of nodes which are not linked to any parent node at upper levels. The union of the receptive fields of these nodes will generate a partition of the input image. It must be noted that these root nodes can be located at the different pyramid levels, i.e. they are selected at different scales. Among these root nodes, the set of keynodes will be chosen. Two constraints are required to be a keynode: its receptive field must not be in contact with the image border, and its colour must be different from the colour of its neighbours. This second condition is always satisfied by a root node, but imposing a minimum value to this contrast measure the algorithm looks for region locations and scales that can be repeatably associated under differing views of the same object. That is, among the image regions associated to the root nodes, the 'maximally stable' ones are those corresponding to nodes whose colour is very different to the colour of their neighbours.

Once the set of visual features has been chosen, each region is normalized geometrically using the covariance matrix. The aim is that the covariance matrix of the transformed region will be equal to the identity matrix. This is achieved by transforming every region pixel by the inverse of the covariance matrix of the original region [18]. Assuming local planarity of the detected region, this geometric normalization, together with the position of the image centroid, provides a rotation-variant measurement of the image. Therefore, if one also assumes that the geometric changes induced by the camera's motion can be described by an affine transformation, one will need to represent the image region by a rotationally invariant descriptor to achieve a viewpoint invariant description. This descriptor will be presented in Section III.

Finally, the image region is normalized photometrically. In this case, it is assumed that the combined effect of different scene illumination and capture system settings can be modeled by affine transformations of individual colour channels. Then, the values of individual colour channels are transformed to have zero mean and unit variance, allowing a patch to be represented invariantly to photometric changes.

III. PCA KERNEL-BASED DESCRIPTOR

Colour histograms have been traditionally employed to provide an efficient image region descriptor, encoding the inner colour distribution of the corresponding set of pixels. Besides, colour histograms can be easily quantized into a small number of bins to satisfy the low-computational cost imposed by real-time processing. On the contrary, colour histograms do not take into account the spatial information. To avoid this problem, the regions can be masked with a kernel in the spatial domain [4].

Specifically, in our implementation, the CIELab colour space has been chosen at the hierarchical grouping algorithm and then also to characterize the colour of the salient regions. Histogram have been quantized in 16 bins, resulting in a descriptor of $16 \times 16 \times 16$ scalar values (4096 values). The descriptor length is then significantly larger than the one of other distribution-based descriptors like SIFT (128 values). This implies more computational time and storage resources. In order to reduce it, we have applied PCA to the kernel-based histograms.

PCA is an approach for dimensionality reduction that determines the directions along which the variability of the data is maximal. For instance, this technique has been applied by Ke and Sukthankar [9] to the normalized gradient patches provided by the SIFT detector or by Mikolajczyk and Schmid [17] to obtain the final GLOH descriptor. PCA is conducted by extracting the eigenvectors of the total scatter matrix of the database S_T, defined as

$$S_T = \sum_{i=1}^{N} (G_i - \bar{G})(G_i - \bar{G})^T \tag{2}$$

where $\bar{G}$ is the mean value of the database of N descriptors G_i.

Eigenvectors W_i and associated eigenvalues λ_i are calculated by solving

$$S_T W_i = \lambda_i W_i \qquad \forall i \in \{1, ..., d\} \tag{3}$$

The transformation matrix is then defined as $\mathcal{W} = \{W_1, W_2, ...W_K\}$, where K is the minimal number of eigen-

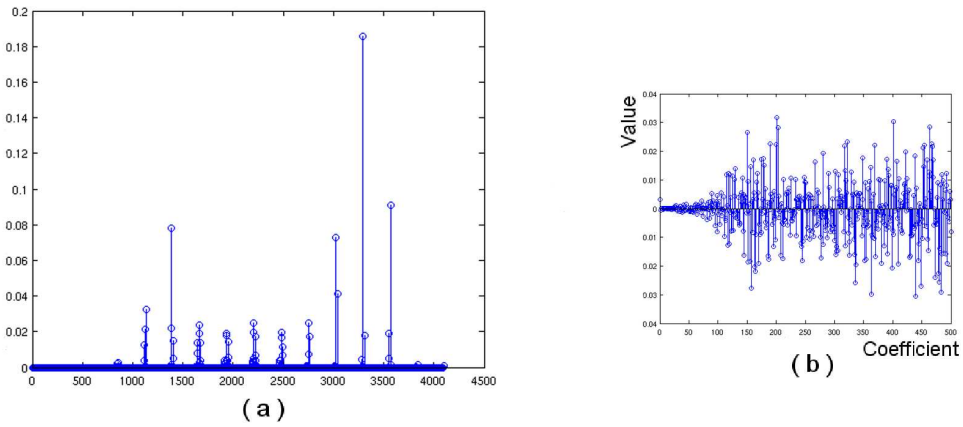


Fig. 4. a) Kernel-based descriptor; and b) PCA projections of the kernel-based descriptor in a).

vectors used to obtain a satisfying representation of the data. Thus, $\mathcal{W}$ is an orthogonal transformation that diagonalises the covariance matrix S_T. In our case, a set of 3100 training samples was used to extract the set of eigenvectors. Then, the compressed feature vector associated to a kernel-based descriptor will be obtained by projecting it onto this set. This projection onto a latent space not only reduces dimensionality but also decorrelates the data. When considering a small database of high dimensionality, this decorrelation can be useful for further encodings, due to the sparsity of data in high-dimensional space.

Fig. 4b shows the feature vectors associated to the kernel-based descriptors depicted in Fig. 4a. In this case, a set of 500 eigenvectors has been chosen to ensure that the projection of the data onto this reduced set covers at least 90% of the data's spread, $\sum_{i=1}^{K} \lambda_i / \sum_i \lambda_i > 0.9$.

Finally, it must be noted that the Euclidean distance between two descriptors can be used to determine whether the two salient regions correspond to the same patch in different images.

IV. Experimental results

A. Evaluation of the proposed segmentation algorithm

The proposed segmentation algorithm has been quantitatively evaluated and compared with other similar algorithms. Three empirical measures have been employed: the Shift-Variance (SV) and the F and Q functions [14]. Shift variance means that the image segmentation produced by pyramid-based algorithms varies when the base of the pyramid is shifted slightly. This is an undesirable effect, so that the SV can be taken as a measure of the quality of a segmentation algorithm. The F and Q functions are measures of the uniformity or homogeneity within the segmented regions together with simplicity in the sense of a relative lack of small holes in the segmentation. Finally, these functions also take into consideration that adjacent regions must present significantly different values of their uniform characteristics.

The SV method compares the segmentation results provided by a given algorithm for slightly shifted versions of the same image. In our case, a window of 128 $\times$ 128 pixels in the centre of the original image has been taken. The segmentation of this subimage is compared with each segmented image obtained by shifting the window a maximum of 11 pixels to the right and 11 pixels down. Thus, there are a total of 120 images to compare with the original one. In order to perfom each comparison between a segmented shifted image I_i and the segmented original image I_{or}, the root mean square colour difference ($\mathrm{RMSD}_{I_{or},I_i}$) has been employed [13]. Then, the SV is expressed as

$$\mathrm{SV} = \frac{1}{120} \sum_{j=1}^{120} \mathrm{RMSD}_{I_{or},I_j} \tag{4}$$

The smaller the value of this parameter, the better is the segmentation results.

On the other hand, the F function is computed as

$$\mathrm{F}(I) = \frac{1}{1000(N \cdot M)} \sqrt{R} \sum_{i=1}^{R} \frac{e_i^2}{A_i} \tag{5}$$

with I being the segmented image, $N \cdot M$ the image size, R the number of segmented regions, and A_i and e_i the area of region i and its average colour error, respectively. The Q function is defined by

$$\mathrm{Q}(I) = \frac{1}{1000(N \cdot M)\sqrt{R} \sum_{i=1}^{R} \left[\frac{e_i^2}{1+logA_i} + (\frac{R(A_i)}{A_i})^2 \right]} \tag{6}$$

with $R(A_i)$ being the number of segmented regions with area equal to A_i. This measure penalizes more rigidly the existence of small regions.

For comparison purposes, five irregular pyramids are employed: the BIP [14], the localized pyramid (LP) [7], the segmentation approach proposed by Lallich et al [10] (MP), the hierarchy of image partitions (HP) [6], and the combinatorial pyramid (CoP) [2]. Four features have been evaluated: the F and Q functions, the SV measure, and the execution time. The images used are a set of 50 images from the Waterloo and Coil 100 databases [13]. The algorithms were run on a 3GHz Pentium IV PC. Table I presents the quantitative results. One appreciates that the shift variance value of the

Fig. 5. a) Reference image and detected features;b)–h) images matched against the reference image. The colour of the ellipses determines if the associated region has been matched to a reference feature (displayed in yellow) or not (displayed in blue).

proposed approach is significantly reduced, providing better results than the rest of approaches. The F and Q values are also improved with respect to the values provided by the original BIP, although they are greater than the ones provided by the HP, the MP and the CoP. Finally, aathough the computation time is slightly greater than with the original BIP, it is still at least ten times less than in the rest of the irregular pyramids.

TABLE I
QUANTITATIVE SEGMENTATION RESULTS: HIERARCHY HEIGHT, NUMBER OF REGIONS OBTAINED, F, Q, AND SHIFT VARIANCE (SV) VALUES, AND EXECUTION TIME.

	F	Q	SV	Time (sec)
LP	743.2	1011.5	30.2	2.75
MP	650.1	818.5	29.3	3.42
HP	670.3	955.1	28.4	4.23
CoP	630.7	870.2	30.5	2.85
BIP	720.2	1090.1	44.1	0.20
Proposed	700.1	950.3	24.3	0.23

B. Testing the approach in an environment mapping framework

The proposed approach has been tested on an ActiveMedia Pioneer 2AT robot. Among other sensors, this robot is mounted with a STH-MDCS stereoscopic camera from Videre Design. This is a compact, low-power colour digital stereo head with an IEEE 1394 digital interface. It consists of two 1.3 megapixel, progressive scan CMOS imagers mounted in a rigid body, and a 1394 peripheral interface module, joined in an integral unit. The camera was mounted at the front and top of the robot at a constant orientation, looking forward. The robot was driven through different environments while capturing real-life stereo images. Images were restricted to 640×480 or 320×240 pixels.

The viewpoint invariance of our approach has been also qualitatively tested. Images of an scene starting from head on (reference pose) and gradually increasing the viewing angle

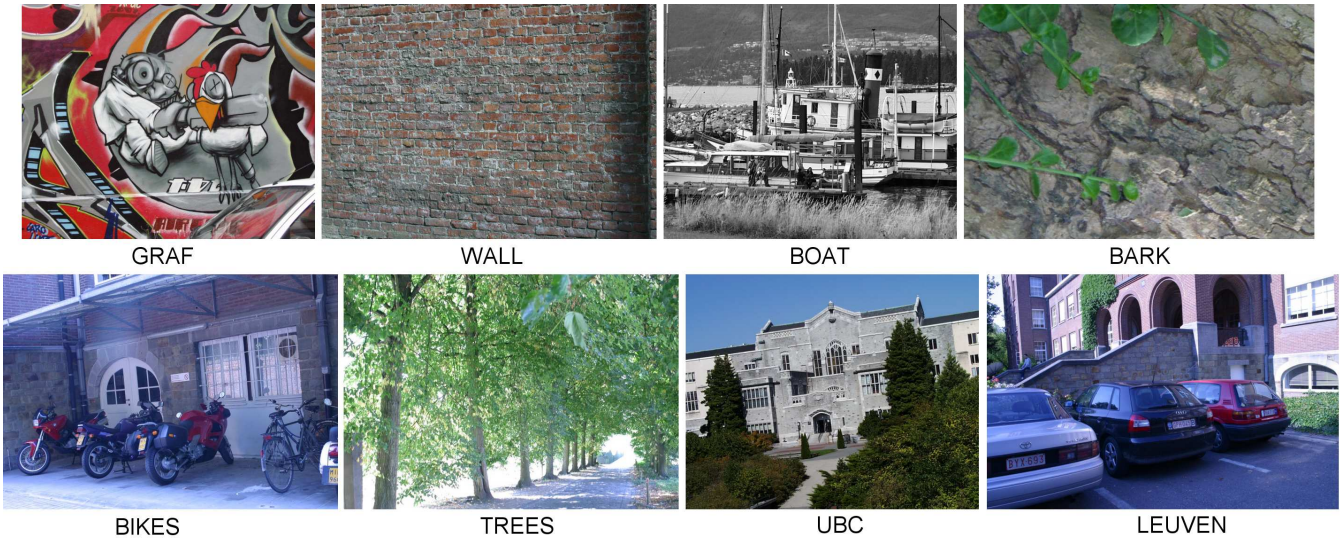


Fig. 6. Image examples of the eight sets used for comparison purposes.

and/or the distance to the reference pose have been captured. Fig. 5 shows one of these experiments, where each visual feature is represented by an ellipse. For each image, visual features are extracted and matched to the features found in the zero degrees reference image (Fig. 5a). A nearest neighbour-based matching strategy has been used, i.e. two regions **A** and **B** are matched if the descriptor D_B is the nearest neighbour to D_A and if the distance between them is below a threshold U. With this approach, a descriptor has only one match. The colour of the ellipses represented in Figs. 5b-h determines if the associated region has been matched to a reference feature (displayed in yellow) or not (displayed in blue). Fig. 11 shows some frames of an experiment conducted in an indoor environment. In this test, detected regions were matched during the trajectory using the same nearest network algorithm. As can be seen, corresponding regions are matched when the same scene is observed from different viewpoints conditions.

Experimental results show that the system can deal with changes in viewpoint up to 50 or 60 degrees and with scale changes of 2 to 2.5. It can be also noted that the number of matches found slightly decreases with increasing scale change.

V. A COMPARATIVE STUDY

Our approach has been also compared with other similar methods employed the protocol described by Mikolajczyk et al. [18]. Images, Matlab code to carry out the performance tests, and binaries of the approaches have been downloaded from http://www.robots.ox.ac.uk/∼vgg/research/affine. Specifically, the database is composed by eight different image sets that represent five changes in imaging conditions (viewpoint changes, scaling, image blur, jpeg compression and illumination changes). These image sets can be grouped into two different scene types: one scene type contains homogeneous regions which present distinctive boundaries (structured scenes), meanwhile the other type contains repeated textures of different forms (textured scenes). As our approach is based on structure cues in images, it is reasonable that it exhibits a superior performance on structured scenes. Fig. 6 shows an example from each image set. It must be noted that the set of parameters employed by the proposed approach has not been modified to deal with the different image sets.

To evaluate the described detector, we use the repeatability score as described by Mikolajczyk et al. [18]. This indicates how many of the detected affine regions are found in both images, relative to the lowest total number of regions found (where only the part of the image that is visible in both images is taken into account). It must be noted that the output for our detector is a set of arbitrarily shaped regions. However, for the purpose of the comparisons using the Matlab code mentioned above, the output region of all detectors are represented by an ellipse. These ellipses have the same first and second moments as the detected regions.

The proposed detector is compared to the difference of Gaussian (DoG) [12], the Hessian-affine detector [16], the *maximally stable extremal region* detector (MSER) [15], the *intensity extrema-based region* detector (IBR) [19] and the Fast-Hessian [1]. For all experiments, the default parameters given by the authors are used for each detector. From Table II, it can be noted that the detectors generate very different numbers of regions, although this also depends on the image type. Thus, some of them provide good results to structured

TABLE II
NUMBER OF DETECTED REGIONS AND COMPUTATION TIMES FOR DIFFERENT DETECTORS FOR GRAF IMAGE (SEE FIG. 6).

detector	Number of regions	Run time (sec)
DoG	1520	0.39
Hessian-affine	1649	2.43
Fast-Hessian	1418	0.12
MSER	533	0.56
IBR	679	9.77
Proposed	147	0.32

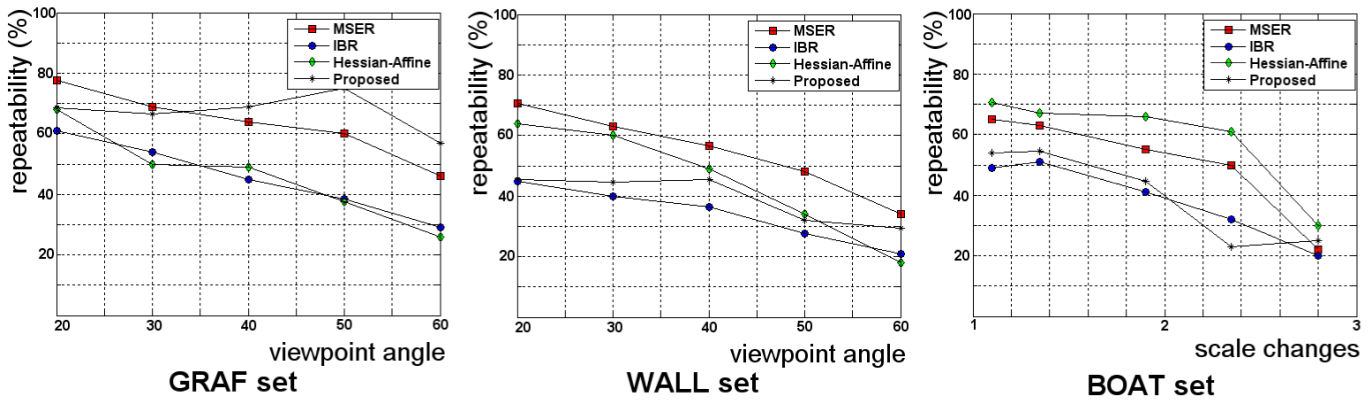


Fig. 7. Repeatability scores for GRAF, WALL and BOAT sequences (see Fig. 6).

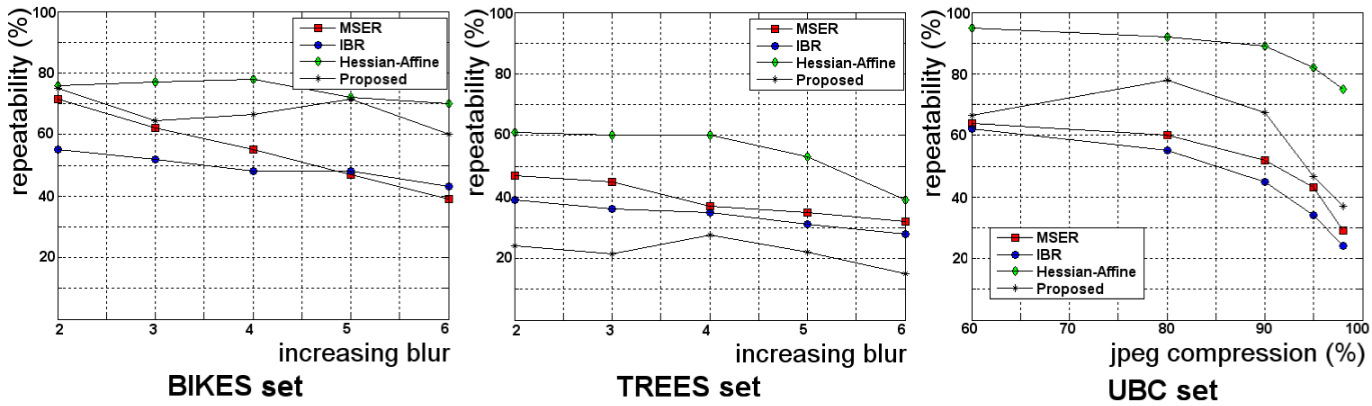


Fig. 8. Repeatability scores for BIKES, TREES and UBC sequences (see Fig. 6).

scenes (e.g. the proposed approach and the MSER) and others to more textured scenes (e.g. Hessian-affine). Table II shows that computation times are also very different. They have been measured on a Pentium 4.2GHz Linux PC, for the GRAF image, which is 800×640 pixels.

The repeatability for six sets of images are illustrated in Figs. 7 and 8. These results show that the proposed detector ranks similar to the rest of approaches when it deals with structured images. In these images, only few regions are detected and the thresholds can be set very sharply, resulting in very stable regions. On the contrary, the scores associated to textured images are significantly bad when compared to the point-based detectors (see Fig. 7, the WALL set).

Finally, the PCA kernel-based descriptor is evaluated using the recall-precision criterion for image pairs, i.e. the number of correct and false matches between two images [17]. Fig. 9 shows the results for three sets of images. Regions have been detected using the proposed approach. Two regions are matched if the distance between their descriptors is below a threshold U. The value of this threshold is varied to obtain the curves (see [17] for further details). Compared descriptors are the SIFT [11], colour SIFT [3] and GLOH [17]. From the results, it can be noted that the PCA kernel-based descriptor performs better than the rest of descriptors. The number of regions is significantly low, and this implies that regions are usually not overlapped. Besides, although the textured scenes contain similar motifs, the regions capture distinctive image variations. For these reasons, distribution-based descriptors like the kernel-based one or the SIFT, exhibit a good performance.

Fig. 10 shows the relationship between the matching accuracy of the proposed descriptor and the dimensionality of the feature space. As expected, increasing the dimensionality of the feature vector results in better accuracy. However, when this dimension exceeds a certain size, the matching accuracy of the algorithm remains approximately constant.

VI. Conclusions

This paper describes an affine region detector whose performance is similar to the current state-of-the art, both in speed and accuracy. To obtain these regions, a hierarchical grouping approach has been performed, generating from the input image an irregular pyramid. Pyramid segmentation algorithms exhibit interesting properties when compared to segmentation algorithms based on a single representation: local operations can adapt the pyramid hierarchy to the topology of the image, allowing the detection of global regions of interest and representing them at low resolution levels. In the obtained hierarchy, the receptive field of a pyramid node is considered as a salient regions if this pyramid node is high-contrasted with respect to its neighbours. This detection is conducted over the different pyramid levels, allowing to detect salient

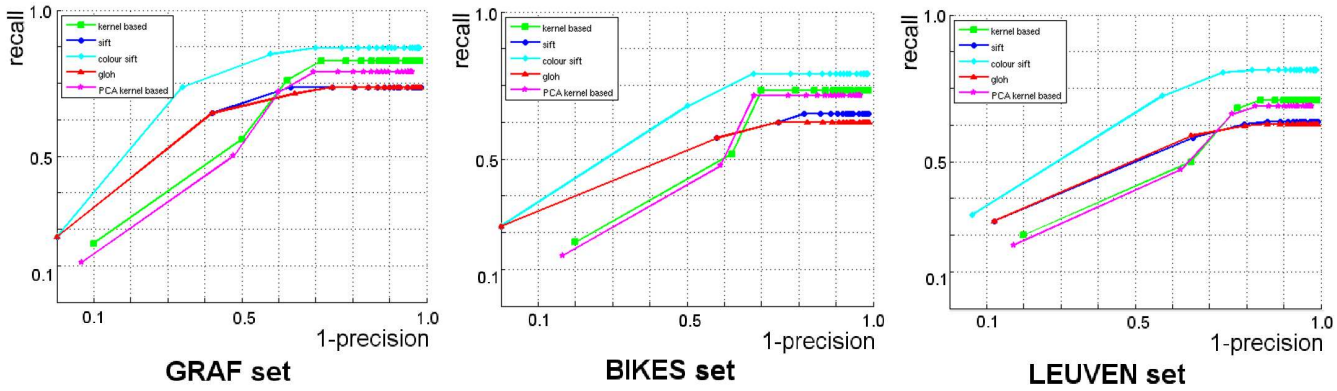


Fig. 9. Recall vs. 1-precision curves for GRAF, BIKES and LEUVEN sequences (see Fig. 6).

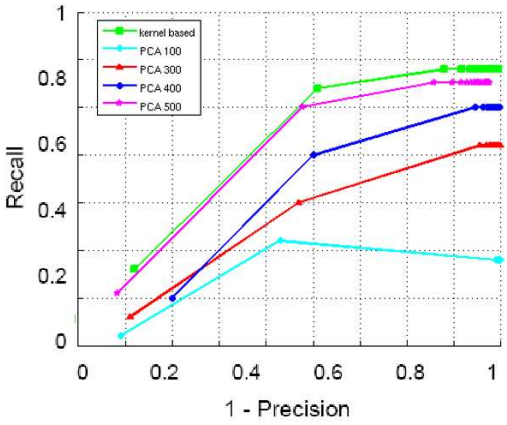


Fig. 10. Performance of the proposed descriptor as PCA dimension varies.

regions at different scales. On the other hand, salient regions have been characterized by a kernel-based descriptor. In order to reduce the large size of this descriptor, we have applied PCA to the kernel-based histograms. The performance of the proposed descriptor is comparable to other similar approaches.

ACKNOWLEDGMENT

This work has been partially granted by the Spanish Ministerio de Ciencia e Inovación (MCINN) and FEDER funds, and by Junta de Andalucía, under projects no. TIN2008-06196 and P07-TIC-03106, respectively.

REFERENCES

[1] H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. In *In: Proc. European Conf. on Computer Vision*, pages 407–417, 2006.

[2] L. Brun and W. Kropatsch. Construction of combinatorial pyramids. In *In: Proc. of Graph-based Representation in Pattern Recognition*, pages 1–12, 2003.

[3] G. J. Burghouts and J. M. Geusebroek. Performance evaluation of local colour invariants. *Comput. Vision and Image Understanding*, 113.

[4] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 25(5):564–577, 2003.

[5] Y. Haxhimusa, R. Glantz, M. Saib, G. Langs, and W. Kropatsch. Logarithmic tapering graph pyramid. In *Proc. 24th German Association for Pattern Recognition Symposium*, pages 117–124, 2002.

[6] Y. Haxhimusa and W. Kropatsch. Segmentation graph hierarchies. In *In: Proc. of IAPR Int. Workshop on Syntactical and Structural Pattern Recognition and Statistical Pattern Recognition*, pages 343–351, 2004.

[7] J. Huart and P. Bertolino. Similarity-based and perception-based image segmentation. In *In: Proc. IEEE Int. Conf. on Image Processing 3*, pages 1148–1151, 2005.

[8] J. Jolion. Stochastic pyramid revisited. *Pattern Recognition Letters*, 24(8):1035–1042, 2003.

[9] Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *In: Proc. Computer Vision Pattern Recogn. 2*, pages 560–513, 2004.

[10] S. Lallich, F. Muhlenbach, and J. Jolion. A test to control a region growing process within a hierarchical graph. *Pattern Recognition*, 36:2201–2211, 2003.

[11] D. Lowe. Object recognition from local scale invariant features. In *In: Proc. 7th Int. Conf. on Computer Vision*, pages 1150–1157, 1999.

[12] D. Lowe. Distinctive image features from scale-invariant keypoints, cascade filtering approach. *Int. Journal Computer Vision*, 60:91–110, 2004.

[13] R. Marfil, L. Molina-Tanco, A. Bandera, J. Rodríguez, and F. Sandoval. Pyramid segmentation algorithms revisited. *Pattern Recognition*, 39(8):1430–1451, 2006.

[14] R. Marfil, L. Molina-Tanco, A. Bandera, and F. Sandoval. The construction of bounded irregular pyramids with a union-find decimation process. *Lecture Notes on Computer Science*, 4538:307–318, 2007.

[15] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. In *In: Proc. British Machine Vision Conference*, pages 384–393, 2002.

[16] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *In: Proc. 7th European Conference on Computer Vision*, pages 128–142, 2002.

[17] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis Machine Intell.*, 27(10):1615–1630, 2005.

[18] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *Int. Journal Computer Vision*, 65:43–72, 2006.

[19] T. Tuytelaars and L. V. Gool. Matching widely separated views based on affine invariant regions. *Int. Journal on Computer Vision*, 59(1):61–85, 2004.

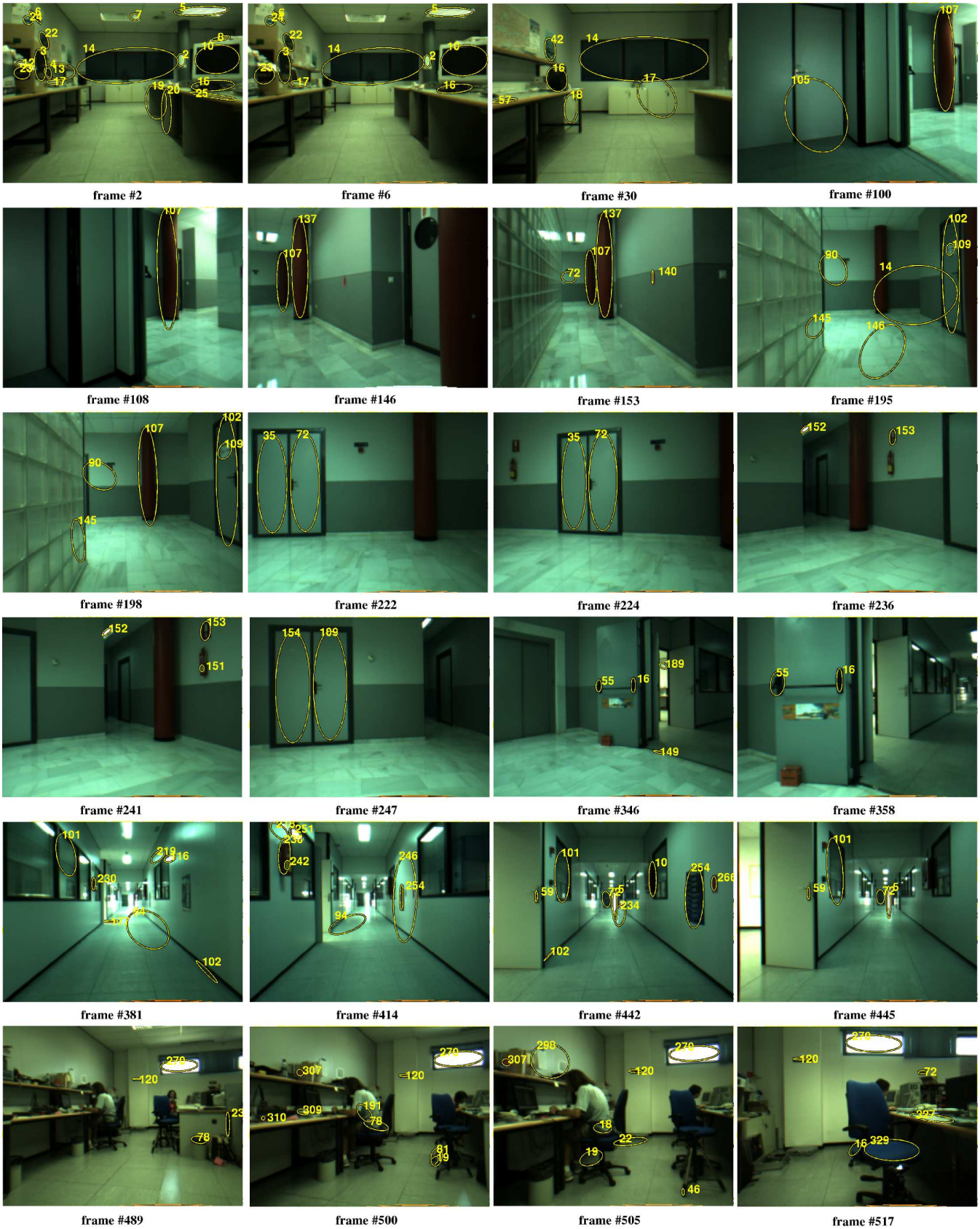


Fig. 11. Experiment in an indoor environment. Detected regions were matched during the trajectory using a nearest neighbour algorithm. It can be seen how corresponding regions are matched when the same scene is observed, e.g. in the following frame sets: (#2, #6, #30), (#100, #108), (#146, #153, #198), (#222, #224), (#236, #241), (#346, #358), (#414, #442, #445) and (#489, #500, #505, #517)