KITE: The Khepera Integrated Testing Environment

Erol Şahin[1] and Paolo Gaudiano[1,2]

[1]Boston University Neurobotics Lab.
677 Beacon St., Boston, MA, 02215, USA

[2]Artificial Life, Inc.
Four Copley Place, Suite 102, Boston, MA 02116, USA

Web: `http://neurobotics.bu.edu`
E-mail: {`erol,gaudiano`}`@cns.bu.edu`

Abstract This article describes an integrated environment developed for the evaluation of localization and navigation algorithms for mobile robots. The integrated environment, named KITE, superimposes spatial information obtained from the robot and its sensors, upon an overhead view of the robot's environment. Using the overhead view as the common coordinate system, KITE creates an ideal test-bed in which navigation and localization algorithms can be visually and quantitatively evaluated in real time. A visual object localization system, which consists of a segmentation and a localization module, developed in KITE is also presented. The segmentation module automatically detects and segments colored objects in real-time, transforming the camera into a simple visual sensor for the Khepera platform. The localization module localizes the objects based on the change in the size of them while the robot is moving. The usefulness of KITE is demonstrated through two experiments. First, the drift in the odometric dead-reckoning of the Khepera is shown. Then the performance of the visual object localization system is evaluated. The visual and quantitative results of this evaluation are presented.

1 Introduction

Localization and navigation are fundamental tasks in mobile robotics. Countless algorithms have been proposed for extracting information about the environment's spatial layout based on information from a robot's sensors (e.g., [1, 2] and many others). However, in most cases it is a difficult task to evaluate the performance of these navigation schemes quantitatively with real robots. This is because in order to verify the validity of sensory data or the performance of a navigation algorithm, one has to know not only the position of the robot, but also the position of the objects that are in the environment.

In some cases the position of the robot is tracked using a number of active or passive beacons mounted in the environment (see [3] for a comprehensive review). The position of objects needs to be derived from a model of the environment created through manual measurements. However, the creation of an environment model through manual measurements is time consuming, it introduces an extra source of error into the system, and makes changes in the environment cumbersome to encode.

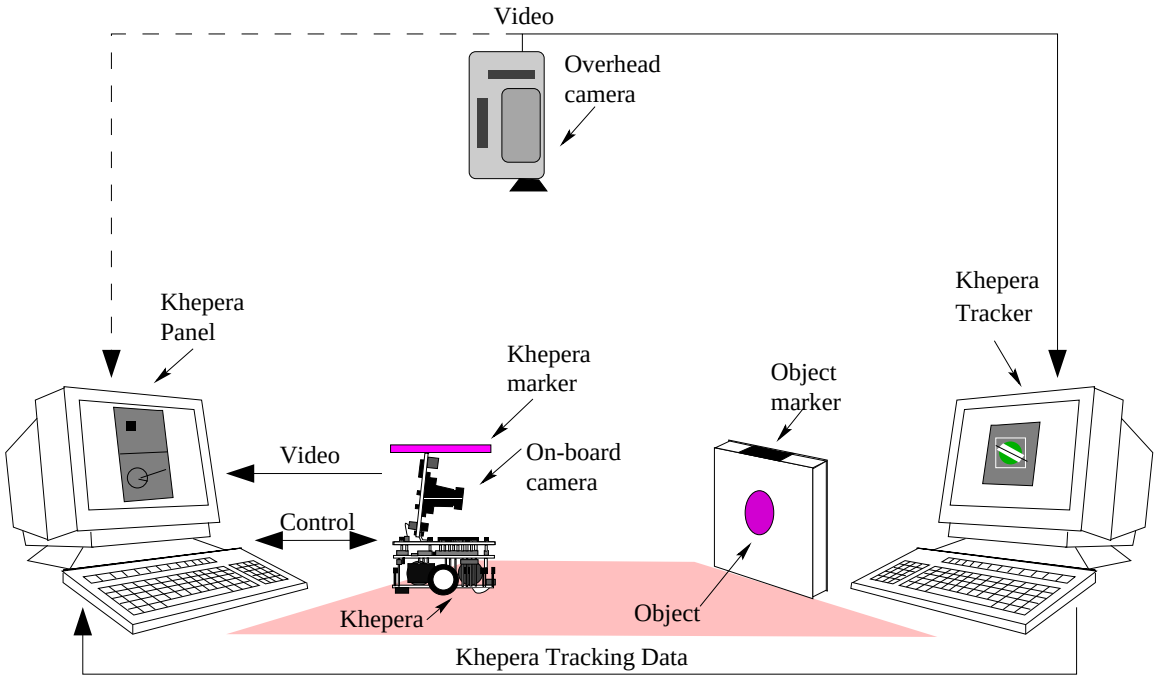


Figure 1. Sketch of KITE. See text for detailed description.

This article describes an integrated testing environment, named KITE (Khepera Integrated Testing Environment), for the evaluation of sensor information and navigation algorithms for mobile robots. KITE combines information from an overhead camera that tracks the mobile robot Khepera (K-Team, SA), with information from the robot's sensors or any algorithm operating on the robot. This makes it possible to obtain immediate visual and quantitative evaluations of the performance of localization and navigation algorithms.

We also introduce a visual object localization system developed in KITE. The system consists of a segmentation and a localization module. The segmentation module automatically detects and segments colored objects in real-time, transforming the camera into a simple visual sensor for the Khepera platform. The localization module localizes the objects segmented, based on the change in the size of them while the robot is moving.

2 Overview of KITE

Figure 1 is a schematic diagram of KITE in which the visual object localization system is developed. The setup consists of two cameras connected to two different computers. The computers communicate through a Local Area Network (LAN). One camera is located over the robot's environment. A colored marker on top of the robot makes it possible for the first computer to track the position and orientation of the robot in real time. We have developed our own *Khepera Tracker* software, which is capable of tracking the robot robustly at full frame rate (over 20Hz). A second camera mounted on the Khepera can be used to perform a variety of vision-based tasks. Images from the on-board camera are digitized and processed on the second computer.

The second computer runs a program we call the *Khepera Panel*, Figure 2 and 3-(a), which carries out multiple tasks listed below:

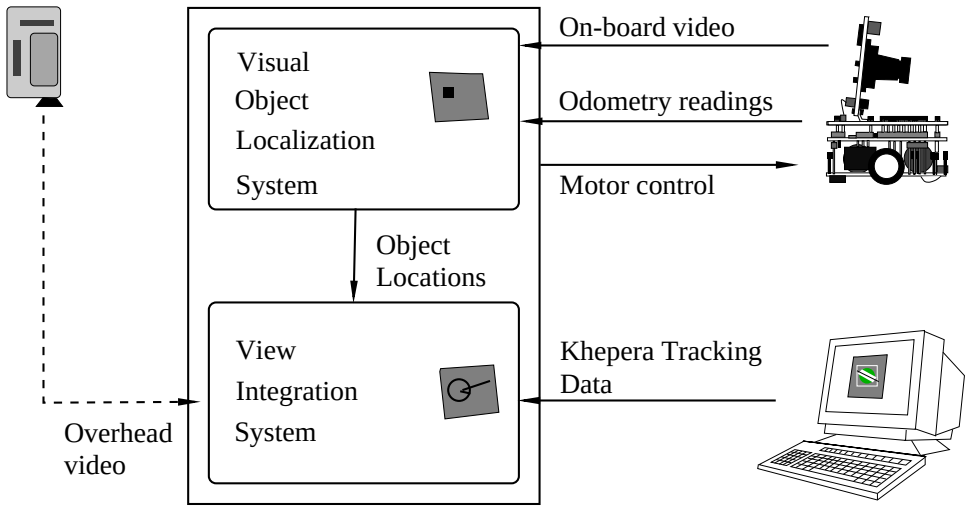


Figure 2. Sketch of Khepera Panel. See text for detailed description.

- Communicates with the Khepera Tracker, receiving the tracked position of the robot through the LAN.
- Communicates with the Khepera, sending out control commands issued by the user or by an algorithm, and receiving the infrared and odometry readings from the Khepera.
- Receives the video data from the robot's on-board camera and the overhead camera.
- Runs the *Visual Object Localization System* described later, which processes the on-board video to extract object localization information.
- Runs the *View Integration System* which creates a integrated view by superimposing different sources of information received or derived on the overhead view.

The Khepera Tracker and View Integration System are the two core components of KITE. Other components of KITE, such as the visual object localization system or the Khepera control component, are peripheral to the core system. They can be omitted or replaced. For instance the visual object localization system can be replaced with any visual algorithm that extracts navigational information. These functionally separate components are implemented in a tightly coupled way due to the synchronization and real-time performance requirements.

The two core components of KITE are described in detail in Section 3 and 4. The visual object localization system is described Section 5. Section 6 reports two experiments carried out in KITE.

3 The Khepera Tracker

The Khepera Tracker tracks the position and orientation of the Khepera using a colored marker placed on the robot, Figure 3-(b). Tracking the robot is based on finding the pixels of the current frame having the color matching the tracking color. Due to the large number of pixels in the image, the search for the pixels matching this color is performed in a small, square shaped region of the image called a *focus window*. After each successfully tracked frame, the focus window is shifted to the current position of the marker and the tracking color is set to the average of the

colors of the matched pixels. This continuous updating increases the robustness of the tracking algorithm in the face of lighting changes.

Tracking the marker is done by comparing the RGB components of each pixel in the focus window with the tracking color. Tracking color is specified by the user with a mouse click after the startup. Both the tracking color and the pixels are RGB normalized, i.e. the RGB color vector length is normalized. This processing step increases the robustness of the algorithm, though it tends to favor the use of primary, saturated colors. When the Khepera marker is lost, the Khepera Tracker enters into a *search mode* in which it scans the full image for the tracking color. Such a mode allows the system to automatically detect the exit and entrance of Khepera into the view. If the tracking color is already known a priori, then the system can be modified to go into the search mode at the startup, detecting the Khepera without the user intervention.

The geometrical center of the matched points is calculated as the current position of the marker, hence of the robot. The orientation of the robot is found by computing the principal axis of the matched points. The ambiguity due to the facing direction of the robot is resolved in the Khepera Panel. This ambiguity is eliminated by assuming that the robot faces in a certain direction 1. Once the initial facing direction is known, tracking the direction is done using the constraint that the change in the direction of the robot will always be less than 90 degrees in between two frames. If the initial facing direction is wrong, it can be reversed by user intervention.

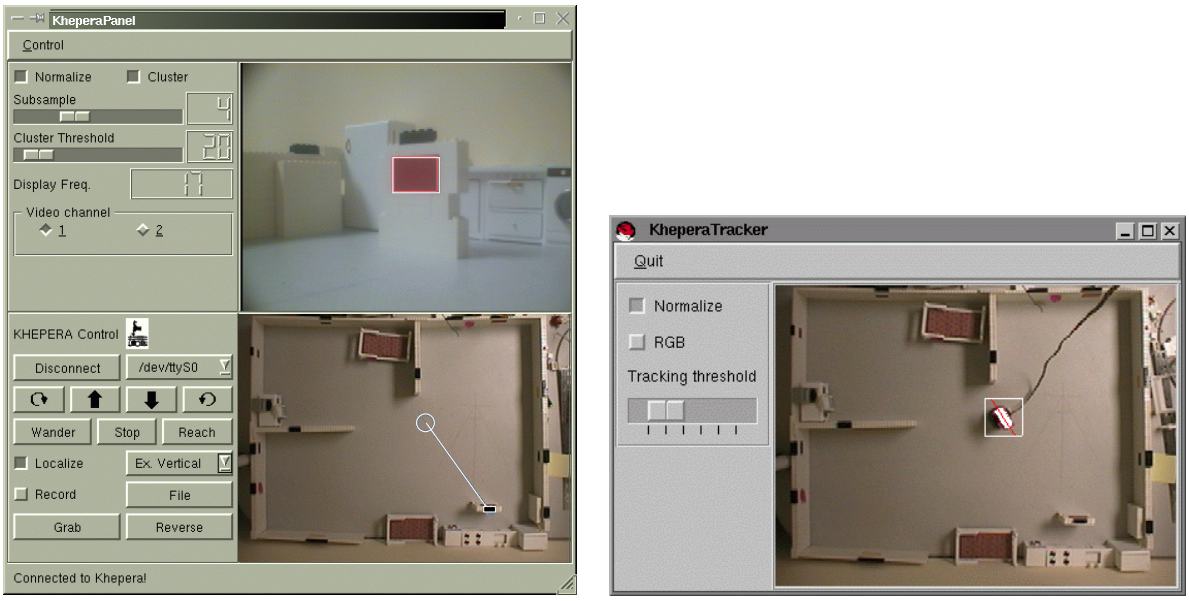
The use of a bi-color Khepera marker would have eliminated the problem of direction resolving. However we chose not to use such a marker since it reduces the number of the pixels in each color blob by half, increasing the error in finding the orientation of robot.

Error in the position and the orientation of the robot is due to the color tracking algorithm and the pixel resolution of the overhead camera. The error in the orientation is pronounced more than the error in the position in most cases. Kalman filtering could provide an optimal estimate of the robot position using the tracking information obtained from the Khepera Tracker and the odometry information obtained from the robot. For instance, such a filtering would average the subsequent position and orientation of the robot when the robot is not moving increasing the accuracy of the tracking information. However this has not been implemented yet.

Similar systems for tracking the Khepera using a overhead camera have been implemented before, [4, 5]. The Khepera Tracker extends these systems by providing a communication outlet through which the tracking information can be sent to other applications running on different (or same) computers. The Khepera Tracker is designed and implemented as a stand-alone tracking tool that can be easily incorporated into other applications.

4 The View Integration System

The View Integration System combines information received by the Khepera Panel into a single integrated view. A sample view created by View Integration System can be seen in the bottom canvas of the Khepera Panel shown in Figure 3-(a). This integrated view is obtained as follows. First, the overhead view of the environment (without the Khepera) as seen by the overhead camera, is grabbed as the background. Odometry readings and the on-board video from the Khepera are then processed by the visual object localization system to estimate the location of objects or other sensory/spatial information in robot-centric coordinates. These estimations are then combined with the tracked position of the robot by Khepera Tracker, to calculate the estimated object locations in the environment. The position of the robot and the estimated



(a) **(b)**

Figure 3. The Khepera Panel is shown in **(a)**. The upper canvas displays the images coming from the on-board video and the output of the segmentation module. The rectangle on the image indicates the object that is detected. The upper part of the control panel displays controls and indicators related to the Visual Object Localization System. The lower canvas displays the integrated view created by the View Integration System. In the lower part of the control panel, the controls for the the robot as well as controls of the integrated view are located. The Khepera Tracker is shown in **(b)**. The white box indicates the current position of the focus window. The matched points are painted in white and the orientation is drawn as a black line in the window.

positions of the objects are then superimposed on the background image. The position of the robot tracked by Khepera Tracker is drawn as a circle, with a small line indicating the facing direction. The object localization estimations are drawn as rays emanating from the robot. Since the position of each object is also visible in the integrated view, it is possible to visually and quantitatively evaluate the performance of the visual object localization system in real time. As described below, using KITE we are able to run a visual object localization algorithm at nearly 20 frames per second using two Pentium workstations with the LINUX operating system.

5 The Visual Object Localization System

The Visual Object Localization System estimates the range and bearing of objects with respect to the robot. By “object” here we refer to anything that can be segmented from the background on the basis of its color. As we describe below, we have devised an extremely flexible color tracking scheme that can track multiple “blobs” of arbitrary colors.

Given an object that is segmented in the image, its bearing can be easily computed with a calibrated camera by converting the horizontal distance between the center of the image frame and the center of the object projection. The object’s range from the robot is computed by

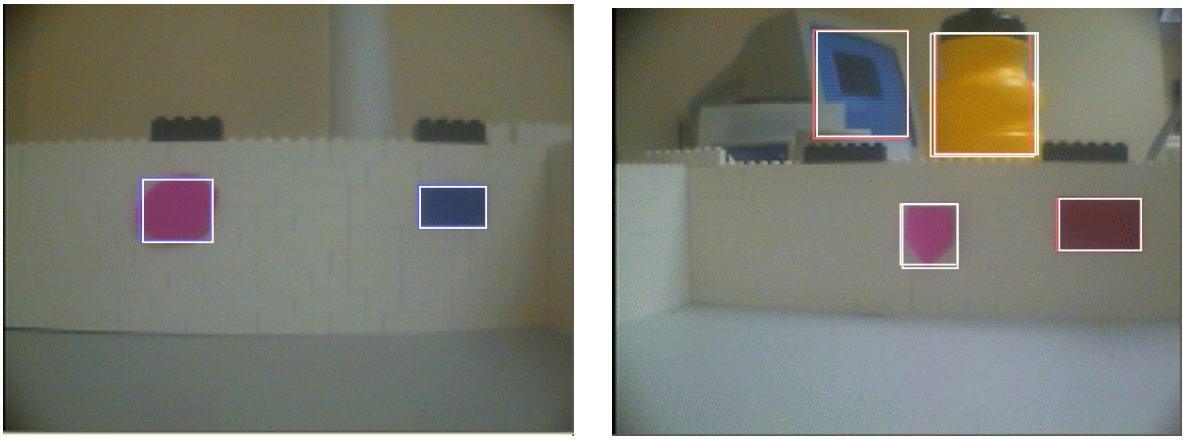


Figure 4. Some sample results of the segmentation module are shown. Although the module was initially designed to detect colored objects such as colored LEGO blocks inserted on predominantly white walls, **(a)**, it can pick up objects, such as the yellow bottle and computer screen, outside its LEGO environment **(b)**.

monitoring changes in the position and size of the object due to known displacements of the robot [6]. Before describing the localization algorithm, we give an overview of the segmentation module we have developed.

5.1 The Segmentation Module

In this project, objects are defined by patches of compact colored regions in the image with no assumptions about their color, size or shape. In particular, blocks of different sizes and colors are inserted into the predominantly white walls. However, as shown in Figure 4, the algorithm works well with other, arbitrary objects. The segmentation module carries out the following four processing steps:

- **Subsampling and normalization:** The raw image (320×240) is subsampled and normalized. Subsampling reduces the processing time, whereas normalization tries to discount changes in the image due to illumination differences. Regions of moderate size (which we are interested in) do not get lost after subsampling. The subsampling ratio used to obtain the results reported in this paper is 4 creating a 16 times reduction in the image data. The subsampling ratio just changes the minimum size of objects that can detected and does not effect the segmentation in any other way.
- **Bounding box clustering:** The pixels in the subsampled image are clustered using their color (RGB value) and position on the image. During this step, each cluster keeps and updates a table which includes information such as the bounding box of the pixels that fall within the cluster. This allows the clustering algorithm to use the additional constraint that a new pixel should be spatially connected to the cluster, if solid blobs are sought. This step creates spatially connected color clusters in the RGB domain.
- **Cluster purging** Three criteria are used to purge "useless" clusters: Compactness, size and position. Compactness measures how dense the pixels of a cluster are in the bounding box of the cluster on the image. It is computed by dividing the number of pixels within

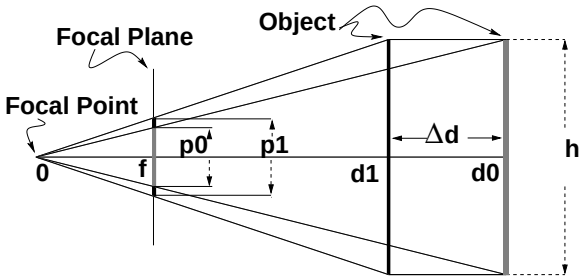


Figure 5. Diagram of the visual looming relationships in a camera-centered coordinate frame.

the cluster by the area of the bounding box of the cluster. Clusters that are not compact enough are purged. Second, clusters with very large or small bounding boxes are purged, since they either correspond to noise, small objects that are not worth tracking or the background. The bounding box of the cluster should be away from the edges of the image by a certain margin. Clusters that correspond to objects which are not fully in the field of view are discarded.

- **Fine tuning the bounding box:** Although the prior steps detect and segment objects, the resolution of the bounding box is low since the processing was done on the subsampled image. To remedy this problem, the bounding boxes of the objects are fine-tuned on the (full-resolution) image.

5.2 The Localization Module

In this article we use KITE to evaluate the results of the *looming algorithm*, a method of localizing objects using a single camera. A more detailed of the looming algorithm can be found in an earlier publication [6]. Visual looming, the expansion of the projection size of an object on the retina, is usually the indication of an approaching object. It can be used to estimate the range to an object using the change in the projection size of the object that results from known robot displacements.

Figure 5 illustrates the geometric relations of visual looming. The figure sketches the case where a camera is viewing an object of size h from two different positions, at distances d_0 and d_1 from the object. Note that it is irrelevant whether the displacement between the two positions is the result of camera movement or object movement. However, when the camera is mounted on a mobile robot, the displacement Δd can be obtained directly through the robot's odometry. As shown below this makes it possible to extract the distance to the (stationary) object.

Given a camera focal length f, the size of the projection of the object onto the focal plane depends on the distance between the object and the camera. In the case shown in Figure 5, p_0 and p_1, respectively, represent the size of the projection of the same object at distances d_0 and d_1. Using similar triangles, it can be easily shown that

$$d_1 = -p_0 \frac{\Delta d}{p_1 - p_0}, \quad d_0 = -p_1 \frac{\Delta d}{p_1 - p_0}$$

where $\Delta d = d_1 - d_0$ denotes the net displacement. Since p_0, p_1 and Δd are all known, the initial and final distance to the object can be computed using this equation. The distance information is then combined with the bearing to localize the object.

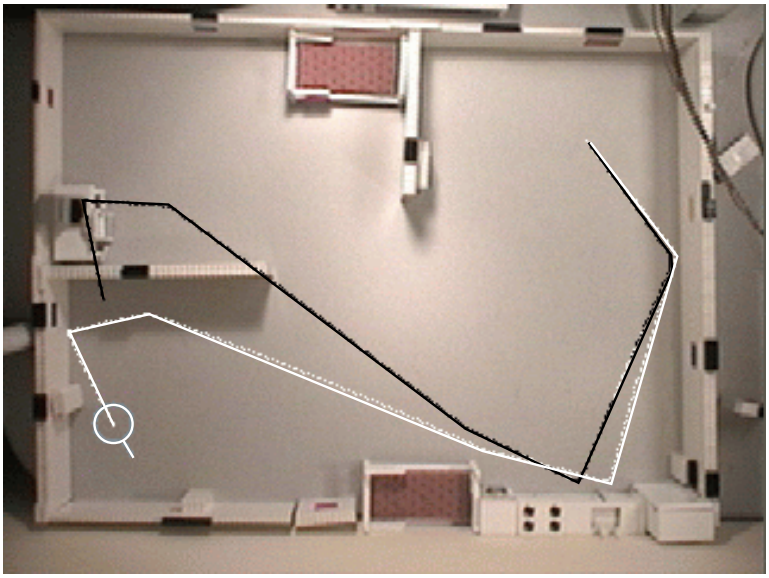


Figure 6. Evaluation of odometry as a positioning method in KITE. Traces of the actual and odometric position of the robot are drawn while the robot is moving. The white points show the trace of the actual robot position, whereas the black points show the odometric position estimate. We observed that the tension or the slack of the Khepera's tether was the main reason for the slippage while the robot is turning and hence the odometric error.

In [6], we have reported some quantitative evaluations and theoretical analyses of the looming algorithm. However, the experimental evaluation required a carefully constrained setup and manual measurements to compare actual and estimated distances. In the next section we described some experimental results obtained with KITE, including an evaluation of the looming algorithm for localization.

6 Experimental results

The purpose of the experiments reported in this section is primarily to show the functionality of KITE. The first example shows the result of error accumulation due to odometry. The second experiment shows the result from the evaluation of the visual object localization system.

6.1 Evaluating Odometry

Odometry is a widely used dead-reckoning method for mobile robot positioning. Although it provides good accuracy in the short-term, its accuracy deteriorates rapidly as the orientation and translation errors accumulates in time.

To show the abilities of KITE, the actual and odometric positions of the robot are traced while the robot is moving. The results, shown in Figure 6, clearly illustrate the accumulation of odometric errors in time. Using KITE, we could for example test odometric slippage under various movement conditions. Furthermore, we could in principle use the visual tracking information to calibrate the odometry of the system in the event of systematic errors.

6.2 Evaluating the Visual Object Localization System

A more impressive example of the usefulness of KITE is shown in 7**(a)**. Here the Khepera is tracked as it moves roughly toward a stationary object. The two circles connected by the black line indicate the initial and final position of the robot. The object consists of a white piece of LEGO wall with a small colored segment embedded in it. The object is marked with a dark piece of LEGO block that is placed on the LEGO wall and is shown as as a small white rectangle with a dark central part. The user clicks on the object marker to point out the actual position of the object in the image. This allows KITE to compute the actual distance between the robot and the object which can then be compared against the localization estimate of the Visual Object Localization System.

The white line emanating from the second circle represents the position of the object relative to the robot as estimated using the looming algorithm with the Khepera on-board camera. Notice that in the picture we only show the initial and final positions, but with KITE we can monitor the robot's position and its estimate of the object's position continuously and in real time throughout the experiment.

In an earlier publication, [6], we had analytically shown that the error in estimating position from looming should decay rapidly as the robot approaches the object. This is confirmed by the results in Figure 7**(b)**, which shows the actual (as measured from the Khepera Tracker) and estimated distance (from looming) as the robot performs the movement shown in part **(a)** of the same figure. The results show that, starting from about 40cm away, the robot's distance estimate falls quickly: after moving 10cm the error is in the order of 20%; after about 20cm the error is in the order of only a few percent, and eventually it approaches zero.

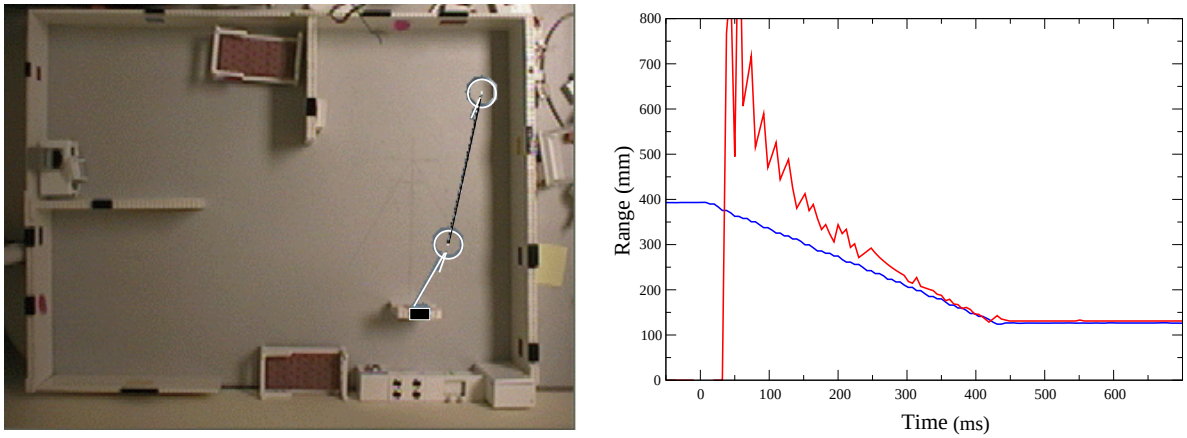
7 Conclusions

In this article we have described KITE, an integrated testing environment for use with the Khepera mobile robot. In particular we showed our results with a system consisting of:

- a stand-alone Khepera Tracker, which can track the current position and orientation of the robot from an overhead camera and send it to any other computer on the network,
- an object segmentation module that can automatically detect and segment colored objects in real-time,
- an object localization module that can localize objects and be used as a range sensor by the Khepera,
- the Khepera Panel, which can be used for the real-time, visual and quantitative evaluation of localization and navigation algorithms on the Khepera.

We think that KITE provides a nice test-bed for the implementation and testing of navigation and localization algorithms with Khepera. The integrated view allows one to see the strength and weaknesses of a navigation or localization algorithm, speeding up the refinement cycle. The quantitative evaluation ability makes it easier to conduct systematic experiments whose results can then be used to obtain statistically valid analyses. Especially in the case of visual algorithms, the Khepera is just as good a platform as any full-size robot. Researchers could test and refine their algorithms using KITE and apply the results to any other mobile robot system.

We also think that the segmentation module transforms the camera into a simple visual sensor that can provide robust, low-dimensional information about the environment. We believe that



(a) **(b)**

Figure 7. Evaluation of the visual object localization system in KITE. See text for details.

this would be a big improvement to Khepera's limited sensory abilities. Finally, the localization module provides the Khepera with a long-range localization sensor which can be used in navigation algorithms.

Acknowledgements. This work is supported in part by the Office of Naval Research and the Naval Research Laboratory through grant ONR-00014-96-1-0772.
E. Ş. received additional support from Starlab NV Research Laboratories, Brussels, Belgium (http://www.starlab.net).

References

[1] **J. Borenstein and Y. Koren**, Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. **5**, pp. 1179–1187, 1989.

[2] **A. Elfes**, Sonar based real-world mapping and navigation. *IEEE Transactions on Robotics and Automation*, vol. 3, no. **3**, pp. 249–265, 1987.

[3] **J. Borenstein, H. Everett, and L. Feng**, *Navigating mobile robots*. Wellesley, MA: A. K. Peters, 1996.

[4] **H. H. Lund, E. de Ves Cuenca, and J. Hallam**, A simple real-time mobile robot tracking system. Technical Paper 41, Department of Artificial Intelligence, University of Edinburgh, 1996.

[5] **C. Chang and P. Gaudiano**, Application of biological learning theories to mobile robot avoidance and approach behaviors. *Journal of Complex Systems*, vol. 1, no. **1**, pp. 79–114, 1998.

[6] **E. Şahin and P. Gaudiano**, Mobile robot range sensing through visual looming. in *Proceedings of the ISIC/CIRA/ISAS*, (Gaithersburg, MD, USA), pp. 370–375, NIST, 1998.