Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 07-003

Autonomous Docking for an eROSI Robot Based on a Vision System
with Points Clustering

Hyeun Jeong Min, Andrew Drenner, and Nikos Papanikolopoulos

January 26, 2007

Autonomous Docking for an eROSI Robot Based on a Vision System with Points Clustering

Hyeun Jeong Min, Andrew Drenner, and Nikolaos Papanikolopoulos
Department of Computer Science and Engineering
University of Minnesota

January 25, 2007

Abstract

This paper presents an autonomous docking system based on visual cues on a docking station. Autonomous docking is essential for large scale robotic teams to be delivered by larger robots, recovered, recharged, and redeployed for continuous operation. Using a computer vision based approach, we identify cues to line up for docking by extracting corner pixels and combining this information with color information. Potential target points are extracted and clustered using Euclidean distance in the image plane. Using these clusters of points the appropriate motion behavior is selected to reposition the robot into the desired position and orientation. This paper will present examples of this implementation using an eROSI robot which uses only vision to navigate.

1 Introduction

There is a great deal of interest in multi-robot teams, specifically with applications in multi-robot collaboration, target tracking, robot platooning, or urban search and rescue. One major aspect of this research is how to deliver and recover robotic teams to the area of interest. In [1] robots follow one another to an area of interest while in [2] larger “mother robots” are used to ship the smaller robots into position. Autonomous docking is necessary to move the robots to a desired orientation and position in order for the robot to resupply or deliver samples. For example, in [2], the robots dock in order to recharge their on-board battery for additional runtime, but this docking may also be used to deliver soil or air samples for analysis which may require larger sensors than all the robots can transport.

Vision-based docking is a challenging problem as there are many environmental variables such as lighting and shadows that make identification of the targets problematic. These problems can be further compounded on smaller robotic systems which lack sufficient on-board processing and must transmit their video for remote processing. This transmission process can add additional noise and distortion to the image. Dealing with this type of noise is critical in identifying the correct position and orientation for docking [3]. However, even with the limitations imposed by noisy imagery, the vision system can be a powerful tool for these types of applications as it provides a wealth of information and is a relatively inexpensive form of sensor.

This paper presents a target detection algorithm based upon the combination of corner detection and color cues. Potential targets are then clustered to increase the likelihood of proper target

identification. Utilizing multiple consecutive images, the method becomes robust to the noise that may be introduced through transmission for remote processing. Figure 1 shows a sample configuration of an eROSI approaching a set of features denoting a docking station.

Once a set of targets is identified, a behavior selection system is activated to control the motion of the robot as it maneuvers to the docking station. As the robot is moving, active perception of the environment is used to generate adaptive behaviors [4]. Adaptive behaviors are necessary as there may be odometry errors due to wheel slippage, terrain features, or imprecise motor control. Thus, behaviors are continually monitored and modified to ensure that the system is robust to noise from the vision system and potentially unreliable motion control.

Figure 1: An eROSI robot and the target for docking.

The approach we present has several novel aspects which make it advantageous over other approaches:

- *Computational Complexity* - Our system reduces computational complexity when handling images since our image processing only extracts around 20 candidate pixels for the target points.
- *Target Representation* - The target object is represented with points. We estimate the point clusters to extract a target object and the method provides the exact information for the corners which will be used for determining the appropriate motion behavior.
- *Target Shape Independence* - The approach can utilize targets of any shape since we use point clustering.
- *Collision Avoidance* - Since the desired targets are known *a priori*, it is possible to estimate depth to the docking position. This in turn can be utilized to avoid collisions with the docking station on suboptimal approach paths.

For our experiments, we will use an eROSI robot, developed at the Center for Distributed Robotics at the University of Minnesota [5]. The eROSI is a two-wheeled, differentially driven miniature robot that is equipped with a miniature camera. However, due to the small size of the

eROSI, it lacks the computational power to process video on-board. Further information on the experimental setup will be discussed in Section 4.

The rest of this paper will be divided as follows. Related literature will be discussed in further detail in Section 2. Section 3 will discuss the framework for alignment for docking along with the specifics of the clustering approaches, selections of behaviors, and pose estimation from features selected in the image plane. Experimental results will be presented in Section 4. Conclusions and future directions for the work will be discussed in Section 5.

2 Related Works

The process of docking can be divided into two main approaches. The first approach or “high-precision” docking utilizes specialized hardware such as laser range finders which generally are very expensive and provide a limited amount of information about an environment. Alternatively, “approximate” docking utilizes lower fidelity sensing capabilities in order to get “close enough” [6]. The “approximate” docking methodology uses sensors such as cameras which can provide more information about the environment, although the precision may not be as high as the approaches utilizing high quality laser scanning.

Vision-based approaches can be applied for finding objects. Vaughn and Arkin [7] used controlled motion performed in a closed-loop and proposed the target extraction method based on a Hough Transform. The method to estimate and execute controlled paths was exploited by Barnes and Liu [8]. They described model-based object recognition and performed deterministic actions. However, using perspective transforms for an external metric is often computationally expensive and can lead to numerical errors.

Arkin and Murphy [9] presented a pattern of actions for docking based on potential fields. Arkin [4] presented a reactive robotic control system to integrate multiple perceptual algorithms. They used a finite state model to deal with perceptual sequencing. Santos-Victor and Sandini [10] presented visual-based behaviors using image measurements through the inverse perspective transform for position and orientation. To reduce computational complexity, Minten [11] presented a low-order complexity vision algorithm which uses two colored regions and color segmentation for image processing.

Our autonomous behaviors are motivated from Arkin’s deliberative architecture, but the specific structures for perceptual processing and deforming state spaces are totally different. We utilize the second derivative to extract potential corners from an image, and use only those points to apply the color matching algorithm. By reducing the number of candidate points, this significantly lowers the computational complexity of the approach. Additionally, the image processing allows for the determination of the present state in our state space and helps to guide transitions between states. This makes action selection more reliable in the presence of noisy imagery and pose estimation.

Outside of the recovery and recharging aspects, autonomous docking can be used for many other applications. Of particular interest is the work of Bereton and Khosla [12] where robots can dock with one another in order to perform analysis and potentially repair one another. This docking was done with a visual docking system where they extracted 8-connected two color regions and defined several states comprised of initialize, approach, center, insert, and retry, in order for the two robots to approach and repair one another.

3 Autonomous Docking

3.1 Framework

Our docking scheme consists of two phases: the *perception* and the *inference* phase. The first phase is for processing consecutive images, and the second phase is for selecting behaviors. Figure 2 shows the framework for our docking alignment.

Figure 2: Framework for docking alignment system.

In the first phase, image processing is used to extract information from the environment in order to determine where in the state space the robot is in order to select the correct motion behaviors. In the specific implementation, the eROSI's lack of memory and processing power requires that the image be transmitted for remote processing on a more powerful machine. After processing, the appropriate motion behavior is relayed through Bluetooth back to the robot. However, on larger, more powerful robots capable of processing their own video, there is no reason that this wouldn't occur on-board.

The images captured generally have enough information to infer the robot's general situation, but these images also include noise from the motors, the transmission of video, and from effects of lighting. As a result, pixels can be "tainted" causing RGB values to be outside the desired target range. Therefore, it is necessary to carefully select hardware and algorithms in order to achieve a system which can react to the environment with limited hardware.

Using combining corner detection and color detection, our system extracts pixel points consisting of the 4 target points. The robot's potential state can be determined from this information as well as the best behavior towards meeting the alignment goals. We will explain the specific methods involved in the second phase which uses the image processing to determine the state the robot is in and how the appropriate behavior is selected in Section 3.2.

3.2 State Space and Point Clustering

The basis of this docking alignment system is to determine what course of action to take by analyzing the environment, extracting features and determining where in the state space the robot is present. Even though the corner detection algorithms reduce the candidate pixel space, there

are still a number of candidate corners detected, some of which correspond to the same target. In our experiment, we use 4 target points instead of a square box or colored regions [12, 3], in order to reduce computational complexity when determining position and orientation.

Figure 3 shows the state space and the relationship between states and behaviors. States are composed from the number of clustering groups that can be extracted from the image at the given position and orientation. Their internal parameter is the threshold for clustering groups, and the external parameters are the lengths and the angles between the clusters of candidate targets, depth, and the number of candidate clusters. Each step includes grabbing images, processing states, executing behaviors, and modifying states and internal parameters.

Figure 3: Sample state space and behavior selection flow diagram.

Each state has the number of candidate clusters; if there are 4 clusters, then the sequence is in Figure 4. After combining corner detection and color classification, our system only detects less than 20 pixels on average. Due to noise inherent to the vision system, it can detect too many candidate pixels. In this case, the system will discard this image and attempt to re-detect. The computational complexity for clustering point groups is $O(a^2)$ where a is the number of extracted pixels and is very small. Also, 4 clustering point groups represent each vertex of a square shape target and we can extract exact corner vertices for the target.

Figure 5 shows before and after clustering the extracted candidate pixels. In this case, there are 8 potential target corners which are then clustered into 4 groups.

The process of clustering pixels representing candidate targets is based on the Euclidean distance between each candidate pixel. After combining the corner detection and color matching algorithms, every extracted pixel is compared with each other. If it is in the threshold boundary, it is then considered part of the same cluster. Determining the correct boundary for thresholding is critical, as the distances between points in the image plane are dependent upon the distance between the camera and the foreground. Therefore, we must estimate the depth at each step and change the

Figure 4: The target points and the sequence for each clustering point.

Figure 5: The example for extracted target pixels and the outcome of clustering them. The numbers represent the sequence of groups used for comparison.

threshold accordingly.

Given that we know the target pattern *a priori*, we can define relative depth by using the ratio of lengths between target points. Since we know the parameters of the target and the camera that is seeking the target, we can establish a minimum distance, d_0 , that the camera must be away from the targets in order to correctly align itself. However, this requires that we are able to successfully identify four potential target point clusters. If we have correctly identified four potential target clusters, we can first estimate depth d at the current time t and then adjust the clustering threshold α :

$$\alpha = \begin{cases} \alpha_{thr} - \frac{\alpha_{thr}|d_t - d_0|}{d_t} & d_t > d_{t+1}, \\ \alpha_{thr} + \frac{\alpha_{thr}|d_t - d_0|}{d_t} & d_t < d_{t+1}. \end{cases} \quad (1)$$

where d_0 is the minimum distance to view all of the potential targets, α is the clustering distance, α_{thr} is the absolute threshold for the clustering distance, and d_t is the estimated distance to the target at time t .

In the case of particularly noisy images, we may be unable to match the points identified with the *a priori* model of the target configuration. This may occur when more than 4 clusters are detected or the 4 clusters that are detected are improperly configured with respect to the *a priori*

model. For example, if the 4 clusters are collinear, there is a problem with the clustering. In this event, we discard the data, and perturb the robot as described in Section 3.3.

3.3 Visual Perception

The goal of this phase is to extract correct clustering groups and estimate some numerical information such as lengths, angles, and depth between the robot and target. The visual information contains very useful data but it usually requires high computational complexity in order to extract reliable features. To reduce the computational complexity and give reliable information, we first apply a corner detection algorithm; namely the Harris Operator [13, 14]. We subsequently match the interesting points with the target color based on the RGB values. Lastly, these pixels consist of clustering groups through the Euclidean distance.

We use the Harris matrix as follows:

$$A = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}, \quad (2)$$

where I is the image being processed. The elements of the matrix A are the second derivatives of an image. The corner is determined by the variation of the second derivative for x and y , and we use

$$r = \det(A) - \alpha \operatorname{tr}^2(A). \quad (3)$$

The larger we allow for the value of r , the fewer corner pixels are left.

For color detection, we first analyze the RGB and HSV values for the target color to train the threshold of RGB values. We extracted the RGB values of each pixel in an image and then converted them to HSV values. HSV is based on intuitive color characteristics such as tint, shade, and tone. Since the RGB values are variant under different lighting conditions, we need to adjust the RGB values. Figure 6 shows the training images used to create the RGB values for the orange color.

Figure 6: Each 320×240 image for training RGB value for the orange color in the different lighting condition.

To construct the target point cluster, we combine the corner detection algorithm and the defined RGB values for the target color. We describe how to cluster pixels in Section 3.2. Now, we need to estimate the external parameters: 4 lengths, 4 angles, the sequence of groups, and the depth. Figure 7 shows the representative groups, 4 lengths denoted by a, b, c , and d , 4 angles represented by α, β, γ , and δ , the sequence of points shown as the numbers, and the depth. The relative computation depends on the variation of depth.

We estimate the following information based upon the number of clusters detected:

Figure 7: The external parameters for state space: Lengths, angles, the sequence of points, and the depth.

- *4 clusters detected* - If the ratio of lengths between the clusters allows it to match the desired target cluster, then we have enough information to calculate the depth. If not, we must discard this information and acquire a new image.
- *3 clusters detected* - If the ratio of lengths between these three clusters allows a partial match to the desired target cluster, we can then detect to maneuver the robot in order to obtain a better view. If the configuration of these three clusters does not allow for a partial match to the desired target cluster, then we must discard the information and acquire a new image.
- *2 clusters detected* - If the two clusters are aligned either vertically or horizontally, we have enough information to attempt to maneuver the robot for a better view. In the case of horizontal alignment, the robot may be too close or too far from the targets causing the remaining two points to be outside of the field of view. Thus, we would adjust the depth of the robot with respect to the target. In the case of vertical alignment, the angle between the robot and the target may need to change in order to bring the other two points into view. If the two clusters are not aligned either vertically or horizontally, we must discard the information and acquire a new image.
- *Less than 2 clusters detected or more than 4 clusters detected* - Here we have either insufficient or spurious data that can not be discriminated and we must attempt to detect again.

In the perception processing phase, when insufficient information is available from a given image, the robot must perturb its position and orientation in order to attempt to gain more information in the next image that it acquires.

3.4 Behavior Selection

As we describe in Section 3.2, we have 4 behaviors: three primitive behaviors for going straight, turning left, and turning right, and one compound behavior for finding target points. The “Finding target points” behavior is to look for other points, and it occurs in the states of *2 clusters detected* or *less than 2 clusters detected*. It accumulates the total degrees turned by the robot at the beginning of the behavior. If a robot is able to properly identify the target points then the total degree would be re-initialized to 0. A robot looks for points within the limited field of view. That’s because visual docking is based on minimizing movements in order to expedite the identification time. Moving a large number of degrees to the right or left in a single step may cause the robot to overshoot the view of the target points, especially in cases where the robot may have a foreshortened view of the points.

Figure 8: Raw image, after detecting the corner pixels, and after clustering the pixels.

For the primitive behaviors the moving distance is $5mm$ and the turning degree is 1 or 3. To look for points the robot needs to expand its search. To avoid collision with the target object, we keep the minimum depth between the robot and the target object. The strategy to select the primitive behaviors is as follows:

1. Keep the target points in the center of the image, and
2. Check “mission done” which is based on the length of the horizontal line.

Lastly, we have the last phase to align a robot to the desired position and orientation. Our strategy keeps the target points at the center of the image, but it sometimes steps aside a little bit.

4 Experimental Results

For the experiments, we used an eROSI robot which has a camera, an analog transmitter for sending video information, Bluetooth hardware for receiving movement commands and communicating sensor readings, and differentially driven wheels. The target object consists of 4 orange square marks. The configuration of the orange square marks coincides with the shape of the docking bay that is being developed and is discussed in more detail in [15]. The eROSI has limited computational power, so the main video computation is done off-board on a PC that has a 1.8 GHz Pentium with 512 MB of RAM.

To see the result for extracting candidate pixels, we saved the image files for the corner detection and the matched pixels. It is depicted in the Figure 8. The first image represents the original image for the target as the robot approaches. The second image is after extracting the corner pixels and the parameters for α and r are 0.04 and 200000, respectively. The last image file shows the result for candidate points which will be used for clustering. In this case, each square target detected 2 to 6 pixels.

Figure 9: Candidate pixels after combining corner detection and color matching.

We also analyzed the number of extracted candidate pixels in each step with results shown in Figure 9. The average is around 10, the minimum is 1, and the maximum is 18. On a rare occasion, it may detect up to 40 points when the transmission from the robot to the receiver is especially noisy.

Figure 10: The whole trajectory for docking.

Figure 10 shows the trajectory for a trial with around 15 steps. There are three different cases in the figure. The first case is represented by red double circles, (1,2,3,9, and 10), and in that case a robot spends more time to decide on its behavior. That's because of processing unreliable image frames in the case of points 1,3 and 9 or just taking a turn the case of 2 and 10. In these short steps, a robot selects "Go Straight" for steps 4,5,6,7,8, and 11. The last position marked with a blue circle and square box represents the last stage to fit it into the desired position and orientation.

5 Conclusions and Future Work

In this paper, we presented a visual alignment system for docking a mobile robot. The mobile robot has limited resources and relies on remote processing of the visual data. Since this type of

hardware system may introduce noisy information, we need to infer what situation the robot is in. Our image processing approach combines corner detection and color matching along with clustering of the candidate points to give flexible execution and low computational complexity.

We have also shown autonomous behaviors for inferred situations or states. Even though the software gives the same commands at every step, there may be some non-deterministic movement resulting from the presence of small obstacles, low power to the motors, or other unpredictable causes. Thus, a robot may lose its way or pass through the location which can detect exactly 4 target points. Our behavior selection can overcome this problem and finally finish the mission for docking.

For future works, we are going to take into account dynamic situations such as multiple robots that want to dock at the same time with the mother ship. We will also consider different target colors by utilizing an adaptive method.

Acknowledgements

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. This work has also been supported through NSF grants #IIS-0219863, #CNS-0224363, #CNS-0324864, #IIP-0443945, and #CNS-0420836.

References

- [1] B. Jung and G. S. Sukhatme, "Tracking multiple moving targets using a camera and laser rangefinder," University of Southern California, Institute for Robotics and Intelligent Systems (IRIS) TechnicalReport 01-397, 2001.
- [2] A. Drenner and N. Papanikolopoulos, "Docking station relocation for maximizing longevity of distributed robotic teams," in *Proc. of the 2006 IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 2436–2441.
- [3] R. R. Murphy and R. C. Arkin, "Adaptive tracking for a mobile robot," in *Proc. of 5th IEEE Int. Symposium on Intelligent Control*, 1990, pp. 1044–1049.
- [4] R. C. Arkin and D. MacKenzie, "Temporal coordination of perceptual algorithms for mobile robot navigation," vol. 10, pp. 276–286, June 1994.
- [5] M. Walter, M. Anderson, I. Burt, and N. Papanikolopoulos, "The design and evolution of the eROSI robot," in *To Appear, Proc. of the 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007.
- [6] N. Barnes and Z.-Q. Liu, "Fuzzy control for active perceptual docking," in *The 10th IEEE Int. Conf. on Fuzzy Systems*, 2001, pp. 1531–1534.
- [7] D. L. Vaughn and R. C. Arkin, "Workstation recognition using a constrained edge-based hough transform for mobile robot navigation," in *Proc. of the Int. Society for Optical Engineering*, vol. 1383, 1991, pp. 503–514.
- [8] N. Barnes and Z.-Q. Liu, "Vision guided circumnavigating autonomous robots," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 6, pp. 689–714, 2000.

- [9] R. C. Arkin and R. R. Murphy, "Autonomous navigation in a manufacturing environment," vol. 6, no. 4, pp. 445–454, Aug. 1990.
- [10] J. Santos-Victor and G. Sandini, "Visual behaviors for docking," *Computer Vision and Image Understanding*, vol. 67, no. 3, pp. 223–238, Sept. 1997.
- [11] B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire, "Low-order-complexity vision-based docking," vol. 17, no. 6, pp. 922–930, Dec. 2001.
- [12] C. Bereton and P. K. Khosla, "Towards a team of robots with repair capabilities: A visual docking system," in *Proc. of the ISER '00 Seventh Int. Symposium on Experimental Robotics*, 2000, pp. 333–342.
- [13] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. 4th Alvey Vision Conference*, 1988, pp. 147–151.
- [14] P. Kovesi, "Phase congruency detects corners and edges," in *The Australian Pattern Recognition Society Conference: DICTA 2003*, 2003, pp. 309–318.
- [15] C. Carlson, A. Drenner, I. Burt, and N. Papanikolopoulos, "Modular mobile docking station design," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robot Systems*, 2006.