Teaching STAIR to Identify and Manipulate Tools

Deborah K. Meduna

December 15, 2006

Introduction

One of the current areas of research in robotics is in the development of robots which can interact with humans in performing a variety of tasks. In particular, there is a desire to enable robots to use objects as tools to perform tasks. For example, assembly tasks may require the robot to use a screwdriver to tighten screws or a hammer for nailing. In this project, I am concerned with the problem of robotic tool manipulation for the purpose of carrying out higher level tasks. This involves developing a learning algorithm to detect the tip of a tool (the control point), as well as algorithms for incorporating the tool as an extension of the robot system and controlling its subsequent movement.

I performed this research on the STAIR robot in the CS Department. Ashutosh Saxena and others have recently completed work in identifying grasping locations on generic objects [1]. Their algorithm enables STAIR to pick up most objects (cups, dishes, pens, tools, etc.) within its field of vision. The goal of my project is to extend this work to allow STAIR to manipulate tools once it has picked them up with the grasping algorithm. One long-term application being considered is to enable STAIR to construct a bookcase from IKEA, given a set of simplified tasks.

1 STAIR Hardware

The STAIR robot consists of a robotic arm and sensor network mounted to a Segue motion platform. It is shown in Figure 1. The vision sensor used for the project is a Focus Robotics stereo camera with pixel resolution of 768x480. I installed the camera and developed code to extract depth information from the images which I then used to calibrate the camera images in the robot coordinate system. This was necessary for translating tip estimates from pixel coordinates to robot coordinates.

Figure 1: STAIR

The STAIR arm consists of 4 joints which are motor controlled to rotate about single axes. Pre-existing code controls movement of the arm to specified locations with respect to the arm base.

2 Tool Tip Identication

The first step in allowing the robot to move and position a given tool is to locate the tool tip - the control point - with respect to the robot. Since the grasping mechanism has already been developed on STAIR, my specific goal was to identify the location of a tool tip given STAIR's image of the tool once it has been grasped by the arm. I focused the tool tip identification process on a single tool: screwdrivers.

2.1 Logistic Regression Algorithm

To implement tool identification, I used a weighted logistic regression algorithm. Since the tool tip size is relatively small in an image, the number of positive training examples is far less than the number of negative training examples. Adding a weighting parameter, $\alpha > 1$, to positive examples serves to partially offset this imbalance. The resulting classifier is

[1] Learning to Grasp Novel Objects using Vision, Ashutosh Saxena, Justin Driemeyer, Justin Kearns, Chioma Osondu, Andrew Y. Ng, 10th International Symposium of Experimental Robotics, ISER 2006.

given by:

$$\hat{\theta} = argmin \sum_{i=1}^{n} w^{(i)}(y^{(i)} - h_\theta(x^{(i)}))^2 \quad (1)$$

$$where \quad h_\theta(x^{(i)}) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \quad (2)$$

$$and \quad w^{(i)} = \begin{cases} \alpha & , y^{(i)} = 1 \\ 1 & , y^{(i)} = 0 \end{cases} \quad (3)$$

Since individual pixels contain little information, the image is reduced to a smaller set of 10 x 10 patches of pixels prior to feature creation. The features used include 9 Laws' mask (relates orientation properties) and 6 texture gradient (includes edge detectors) features for a total of 15 features per image patch. These are shown visually in Figure 2. In addition, I use a canny edge detection feature with value 1 if an edge is found within the patch and value 0 otherwise. This is meant to account for the fact that the screwdriver tip will likely be on or adjacent to an edge in the image.

Figure 2: Image Texture Features. The first nine images are Laws mask filters, followed by the texture gradient filters.

In addition to the 16 image features described above, the algorithm appends the features for the neighboring patches to each patch feature vector in order to gain more global information. The appended features are associated with the patches to the top, bottom, left and right of the current patch. Altogether, the feature vector for a given patch contains 80 features: $x^{(i)} \in R^n$ for $i = 1..m$, where $n = 80$. The number of training examples is given by $m = (rcN/q2)$ where r and c are the number of rows and columns in the image, N is the number of total training images, and $q = 10$ is the patch size. In general, $m >> n$ so the θ parameters can be uniquely determined from gradient descent.

2.2 Consolidating Tip Estimates

The classifier described above often positively classifies multiple patches within a new image. It was thus necessary to develop a method for consolidating the estimates into a single, best estimate of the tip location.

2.2.1 Selective Classified Patch Removal

The first step in estimate consolidation is selective removal of classified patches. Estimated patches were removed if:

1. patches were far away from all other estimated patches (isolated)
2. patches were far away from edges (as defined from canny edge detection)
3. patches had no neighboring patches BUT there were other classified patch clusters in the image

These rules remove many false classifiers, always leaving at least 2-5 classified patches in the image. Figure 3 shows an example of the removal process results.

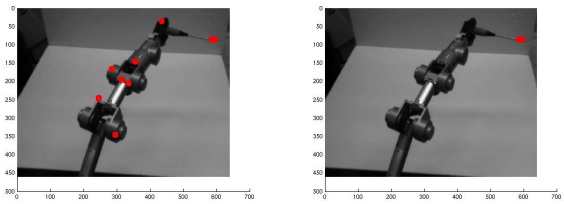

Figure 3: Classifier Patch Removal. Red squares indicate positive classified image patches

In this particular example, all patches were removed except for the two on the tool tip because the two patches are clustered.

In the future, the number of false classified patches can also be reduced by limiting the image search space. Since the tool is assumed to be grasped by the arm, the gripper location can be used as a prior to reduce the image space to a box around the gripper capable of containing all possible sizes and orientations of known screwdrivers. This would alleviate false classifiers far from the screwdriver location and improve overall algorithm performance.

2.2.2 Screwdriver Edge Detection

Once the number of classified patches has been reduced, another algorithm uses the remaining patches to identify the screwdriver edge in the image. Canny edge detection is used to select edges in the image. Then the edge is selected which is closest to the remaining classified patches. Examples of this result are shown in Figure 4.

The image on the left shows the resulting screwdriver edge classification for the images in Figure 3. Here,

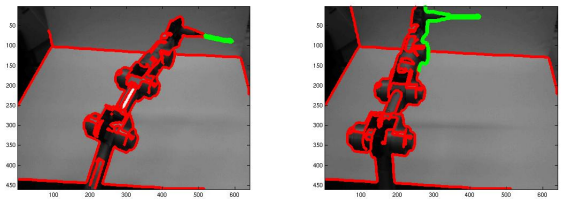

Figure 4: Screwdriver Edge Identification. Red lines indicate identified edges. Green lines indicate identified screwdriver edge.

the identified edge is defined just along the screwdriver end. The image on the right shows a case where the identified screwdriver edge includes some of the robot arm edge as well. This happens as a result of the canny edge detector choices in breaking up distinct edges. Adjusting parameters in the edge detection could help reduce some of this effect.

2.2.3 Finding Tip from Screwdriver Edge

Once an edge is identified, the next step is to select the screwdriver tip from the edge. Several paths were explored for this purpose but have not yet been successfully implemented. The two most promising directions are:

1. Extract line segments from the identified edge using the Hough Transform, then choose the tip as the end point on the 'best' line. Unforunately, the longest line segment is often not the line with the tip, requiring a more sophisticated algorithm.

2. Estimate the curvature of the identified edge line and choose the tip as the point corresponding to maximum curvature. The difficulty with this technique is in accurately calculating the curvature.

These methods and others are still being explored.

3 Results

The classification algorithm described in the previous section was tested on two sets of data: one with 150 images taken in uncluttered background, the other with 65 images taken in cluttered background. All images were taken with the screwdriver in the robot arm grasp. Examples of cluttered and uncluttered images are shown in Figure 5.

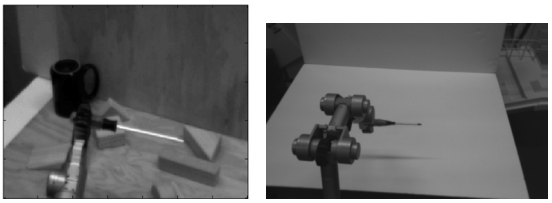

Figure 5: Sample images. Left is cluttered background, right is uncluttered.

In order to characterize the performance of the classifier, I performed K-fold cross validation with a k of 10. The results are shown both before and after the patch removal step in the following tables:

Table 1: K-Fold Cross Validation Results - No Patch Removal

		FN	FP	TL	SE
Uncluttered Background	Test	0.58	4.1	0.77	0.67
	Train	0.47	3.74	0.8	
Cluttered Background	Test	0.84	8.66	0.47	0.52
	Train	0.52	8.13	0.66	

Table 2: K-Fold Cross Validation Results - With Patch Removal

		FN	FP	TL	SE
Uncluttered Background	Test	0.67	3	0.69	0.86
	Train	0.55	2.52	0.72	
Cluttered Background	Test	1	3.24	0.43	0.74
	Train	0.63	2.04	0.58	

Here,
FN = Avg False Negatives/Image
FP = Avg False Positives/Image
TL = Fraction of Images which Correctly Classify Actual Tip Location
SE = Fraction of Images which Correctly Identify Screwdriver Edge

As expected, the average number of false positives is reduced for both data sets when the patch removal process is included. However, the ability of the classifier to correctly identify the trained tip location is reduced. This worse perfomance is partially artificial. The patch removal process sometimes removes patches right at the tip because those patches are often isolated. At the same time, however, the patch removal process improves the accuracy of the screwdriver edge detection significantly for both data sets (by about 18%). As expected, the uncluttered background data set performed better than the

cluttered background data.

Finally, the similarity of the test and train results for the uncluttered data set indicates that the algorithm has relatively small variance. This also indicates that in order to reduce the errors further, I will likely need to include additional features. Additional features could include other edge detector algorithms or other image characteristics such as hue.

4 Tool Incorporation on STAIR

In order to use the tool tip estimate to manipulate the tool using STAIR, the estimate is first used to compute the tool transformation vector. The resulting vector is then incorporated into the prexisting STAIR control code to move the tool to a specified location.

4.1 Tool Transformation Vector

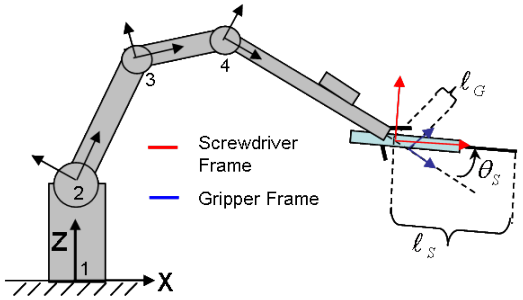

Figure 6: Diagram of Screwdriver in Arm

The tool transformation vector gives the tool tip position in the gripper coordinate frame. Once in gripper coordinates, the tool tip position can be determined with respect to any part of the robot arm through pre-existing coordinate transformation matrices. This then allows for control of the tip position using pre-existing code.

The screwdriver and gripper coordinate frames are illustrated, along with the robot arm, in Figure 6. In terms of the parameters in this figure, the tool transformation vector is given by:

$$\vec{x}^g_{tip} = \begin{pmatrix} l_s cos(\theta_s) + l_g \\ 0 \\ l_s sin(\theta_s) \\ 1 \end{pmatrix} \quad (4)$$

If T_{Bg} is the transformation matrix from the gripper frame to the arm base frame, then $\vec{x}^B_{tip} = T_{Bg}\vec{x}^g_{tip}$.

The tip location and the gripper location with respect to the base can then be related as follows:

$$\vec{x}^B_g = T_{B4}(l_{4g} \quad 0 \quad 0 \quad 1)^T = T_{B4}\vec{b} \quad (5)$$

$$\vec{x}^B_{tip} - \vec{x}^B_g = T_{Bg}\vec{x}^g_{tip} - T_{B4}\vec{b} \quad (6)$$

In Equation 6, T_{B4} is the transformation matrix from the arm base to the 4th joint (right before the gripper), and l_{4g} is the length of the arm segment from joint 4 to the gripper. The resulting expression for the tool vector is:

$$\vec{x}^g_{tip} = (T^T_{Bg}T_{Bg})^{-1}T^T_{Bg}[\vec{x}^B_{tip} - \vec{x}^B_g + T_{B4}\vec{b}] \quad (7)$$

As the equation indicates, in order to find the tool transformation vector, all that is necessary is a set of gripper and tool tip positions with respect to the base. I programmed a pre-planned trajectory to move the grasped tool through 20 distinct positions and orientations. At each point, the gripper position is stored and the tool tip position with respect to the base is estimated using the identification algorithm from Section 4. The estimated tool tip position in the image plane is transferred to a position with respect to the arm base using the camera to robot transformation matrix (see Section 1).

4.2 Commanding Tool Movement

Once the tool transformation vector has been determined, it can be incorporated into the pre-existing STAIR control code. The current code commands movements to STAIR's arm joints based on minimizing a cost function which exponentially penalizes for arm joints being too close to obstacles and walls and being far away from the goal. The goal is a 3D position in space, relative to the robot arm base, specifying the gripper location.

The current incorporation of the tool into this positioning system uses the tool vector to specify the tool tip position as the goal and penalizes for distance between the goal position and the tool tip position at each iteration. This implementation provides accurate tool tip positioning to within a radius of 5 mm of the goal. However, it does not yet allow for specifying tool orientation in the final position state. This will be a future extension.

Figure 7 shows an example of moving the tool to a desired location. Here the tool goal and tool vector

were hard coded ahead of time, but the positioning system was all automated. Once the tool tip estimation algorithm is complete, the system will be able to seamlessly transition from tool calibration to tool positioning without external input.

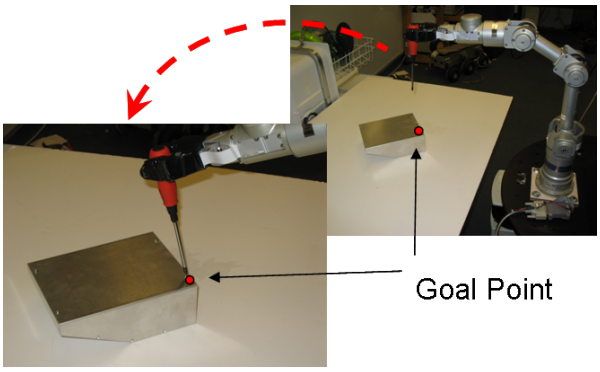

Figure 7: Moving tool tip from initial position to desired location

5 Conclusions

The goal of being able to identify a grasped tool's orientation and move it to a desired location was almost achevied. Many of the sub-steps proved very successful, including the tool transformation vector calculation and the incorporation of the tool into the robot positioning system. The primary missing link is the ability to predict a single tool tip location from an identified screwdriver edge. The selective patch removal and the screwdriver edge identification algorithms seem to give promising results in terms of progressing towards a single tip estimate. The final step in finding the tip from the screwdriver edge will likely be a combination of hough transform and concavity analysis.

In addition, work can be performed on improving the performance of the algorithm for cluttered and unstructured environments. Much of the algorithm performance improvement can be explored by analyzing the effect of the numerous parameters in the system. These include:

1. α, the weighting parameter for the regression algorithm
2. q, the patch size for the feature creation
3. the canny edge detection parameters
4. the distance parameter in the patch removal process corresponding to the threshold at which patches are kept

Adjusting these parameters will give further insight into the problems arising in each step of the algorithm.

Finally, once the process has been acheived for the screwdriver, further extensions of this work can be done to include other tools such as hammers, pliers, etc.

Acknowledgements

I would like to acknowledge and thank Ashutosh Saxena and Andrew Ng for their help on all aspects of this project including primarily algorithm development and project direction. I also thank Justin Drieymeyer and Morgan Quigley for invaluable hardware and software help on STAIR.