Enabling a Robot to Open Doors

Andrei Iancu, Ellen Klingbeil, Justin Pearson
Stanford University - CS 229 - Fall 2007

I. Introduction

The area of robotic exploration is a field of growing interest and research in the robotics and computer science communities. Ideally, a human would send a robot into an area otherwise too dangerous for the human to enter, such as a smoke-filled building, a structurally-unsound bridge, or a radioactive area. Although robots already exist which can map a building's floor plan, they can only do so provided the doors are all open – a robot that can identify and open doors as it maps a building remains to be developed.

The goal of this project was to use machine learning to enable a robot to identify the location of a door handle in an image, classify the door handle type, and finally open the door. The data set considered consisted of three door handle types – left and right-turn handles and elevator buttons. Several machine learning algorithms were implemented using a variety of different feature sets. For both the identification and classification problems, the best results were achieved using a derived feature set with Principal Component Analysis (PCA) and the Support Vector Machine (SVM) algorithm. Accuracy for identifying a door handle in a scene was approximately 99%, and the accuracy for classifying the door handle type was approximately 99.5%. The algorithms were demonstrated on the STanford Artificial Intelligence Robot (STAIR), which was able to successfully execute the motion of pushing down on a door handle to unlatch it.

II. Identifying a Door Handle in a Panoramic Image

In terms of developing the capability of identifying a single door handle in a tightly-cropped image, we have attempted implementing several different solutions before converging on the use of a binary classification SVM. As an initial cut at the problem, logistic regression, naïve Bayes and SVM were implemented on a data set consisting of approximately 500 25 by 25 images half of which depicted door handles with the rest consisting of other varied building scenery. The pixel RGB values were used as the features. The results were not very encouraging. Naïve Bayes was clearly the most unsuitable algorithm with almost 50% error on most test sets. In retrospect, this was to be expected as the Naïve Bayes assumptions on feature independence would obviously be invalid when applied to image pixels. Logistic regression provided a starting test set error of approximately 25% to 30%, while the SVM's test set error hovered around 15%. Based on this initial round of tests, we decided to pursue the binary classification SVM in more depth. Although multiple implementations of the SVM algorithm were assessed, the software ultimately used was SVM Light developed by Thorsten Joachims at Cornell University. [1]

It should be noted that in all SVM training from this point onward, false positive errors were weighted much more heavily than false negative errors in order to eliminate as many of the spurious positive door handle identifications in a panoramic image as possible. The first attempt at refining the SVM algorithm centered on varying the size of the input images and observing the effect on training/test set error with the pixel RGB values still being used as the features. The image size was varied from 4 by 4 to 25 by 25 and the primary observation from this set of tests was that the training set error would generally be zero for all image sizes larger than 5 by 5 while the test set error would hover around a fixed mean of ~12% to 15%. This seemed to indicate that our algorithms were over-fitting and that we needed to either get more training examples or use a better set of features. We opted to use a set of derived features that were computed from the pixel RGB values and designed to capture three types of local cues: texture variations, texture gradients, and color, by convolving the image with 15 filters (same features used for classification, see Section III for a more detailed description), our test set error dropped slightly so that it now hovered around 9% to 12%. However, even with the derived features, our training set error was still always zero so the over-fitting problem seemed to persist.

To attempt to overcome the over-fitting problem when using the derived features, we resorted to PCA and attempted to remove some of the directions in our feature space that did not seem relevant to our identification problem. The results of the PCA were actually quite interesting in that using the primary components of only the images containing door handles actually provided better results, on the order of 1 to 2%, on the test sets. However, this seems intuitive in that we are primarily interested in correctly identifying door handles and the non-door handle images would clearly contain undesired principal components. To select the optimal filter size when calculating the derived features, i.e. the number of pixels from each image blended together, as well as the optimal number of principal components, an iterative algorithm was used to map out the entire optimization space. The results are shown in Figure 1 and Figure 2 below.

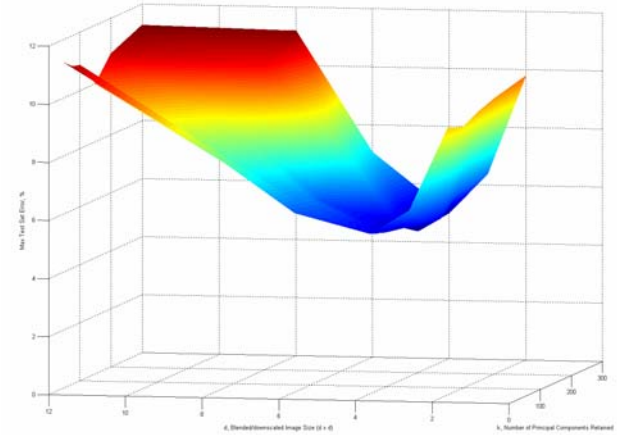

Figure 1. Max test set error vs. filtered/downscaled image size (d) and # of principal components (k).

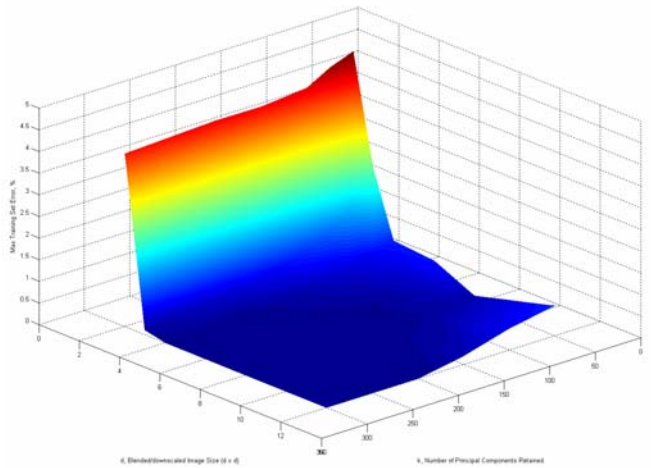

Figure 2. Training set error vs. filtered/downscaled image size (d) and # of principal components (k).

As highlighted by the above data, there is an optimal point with an average error of ~4% where the over-fitting problem disappears and the error is very close to its lowest observed value. This point corresponds to blending the image down to 4 by 4 pixels using the previously mentioned filters then picking the first 200 principal components of the resulting feature set. The result of this was a classifier that, given a tightly-cropped image, could identify whether it contained a door handle with > ~96% accuracy with the vast majority of the misclassifications being false negatives.

To tackle the problem of identifying a door handle in a panoramic image, the identification function sequentially steps through the input image using a variable or fixed size frame that is approximately the expected size of a door handle. The frame size is generally chosen based on laser depth data. The RGB image that is the content of the frame at each step is then passed through the previously developed SVM classifier which classifies the contents as either a door handle or not. This results in a cloud of positive classification hits around each door handle in the image where the centroid of each cloud is returned as a single, confirmed door handle, as shown in Figure 3 below. In the case of multiple clouds, the K-means algorithm is used to calculate the centroid of each cloud.

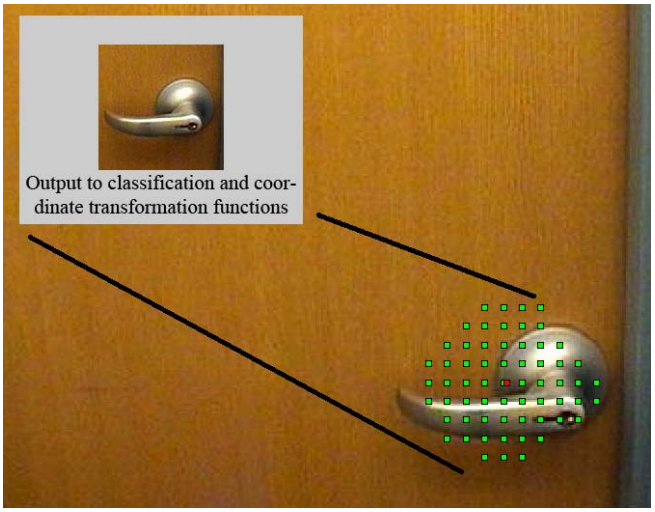

Figure 3. Example of panoramic image parsing on STAIR robot (Green dots = positive classification / Red dot = final output).

Using this particular algorithm, the identification function achieved > ~99% accuracy at identifying a door handle/elevator button in a scene. This accuracy was much higher than the tightly-cropped image identification accuracy due to the weighting of the false positives/negatives in the latter which, when combined with the block parsing of the panoramic image, consistently yielded large clouds only in regions containing an actual door handle.

III. Classifying Door Handle Type

Once the robot has identified the location of the door handle in an image, it needs to identify the handle type to know how to manipulate it and open the door. The output from the door handle identification algorithm is an image cropped around the region containing the door handle. Because the size of the cropped image output from the identification will be variable, it is re-sized to a constant dimension using nearest neighbor interpolation. For a first cut, several machine learning algorithms were tested on an initial training set consisting of only left and right turn handles. Approximately 120 training samples were used. Each cropped image was re-sized to 25x25 pixels and the RGB pixel values were used as the features. Logistic regression had a test error of approximately 20%. Naïve Bayes had a test error of approximately 40%, and SVM had the lowest test error, ranging between 5 and 10%.

The SVM algorithm appeared to give the best results for the two-class training set, so we decided to move forward using this algorithm. We used the SVM Multiclass open source software package created by Thorsten Joachims, Cornell University [2]. SVM was implemented on a three-class training set consisting of approximately 1200 images of elevator buttons and left / right turn handles. Hold-out cross validation was used to compute the average training and test errors. The SVM regularization parameter was varied and the images were re-sized to different values. An image size of 35x35 and a SVM regularization parameter of 170 gave the lowest test error of 2.3%. Our goal was to have classification accuracy of at least 98%. Also, the training error was zero for any image size greater than 10x10. This indicates over-fitting. Reducing the number of features by decreasing the image size gave non-zero train error, but increased the test error to 3.5%.

Based on the experiments, it appeared that improving the errors would require a better and reduced set of features. First the images are re-sized to 50x50; some example training samples can be seen in Figure 4. The set of features were computed from

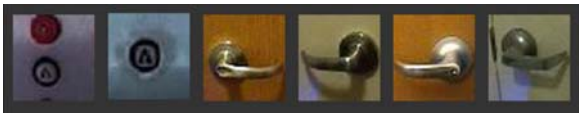

Figure 4: Example 50x50 training samples for door handle type classification.

the pixel RGB values and are designed to capture three types of local cues: texture variations, texture gradients, and color, by convolving the image with 15 filters (9 Laws' masks and 6 oriented edge filters (Fig. 5). The image is segmented into 4x4

patches of pixels and the filters are applied to all 3 color channels for each of the patches for a total of 4x4x45 features per patch. The features for the 4x4 patches are summed over the entire image to give a total of 4x4x45 features per training sample. PCA is then used to extract the most relevant features from this set. The algorithm for feature extraction was provided by Ashutosh Saxena. [3]

Figure 5: The filters used for computing texture variations and gradients. The first 9 are Laws' masks, followed by the oriented edge filters.

Finally, SVM is run on the reduced set of features to classify the door handle as one of the three types. An exhaustive search was used to determine the number of principal components to use as well as what value to use for the SVM regularization parameter to achieve the best accuracy. The SVM algorithm was run 5 times on each set of parameters using different train and test sets to compute the average errors for each run and compute the variance of the errors. Figures 6 and 7 show the results of the training and test errors obtained for varying the number of principal components (k) and the regularization parameter (C). From the figures, we see that there are many values for these two parameters that give test errors at or below 0.5%, with variance of less than 0.3%.

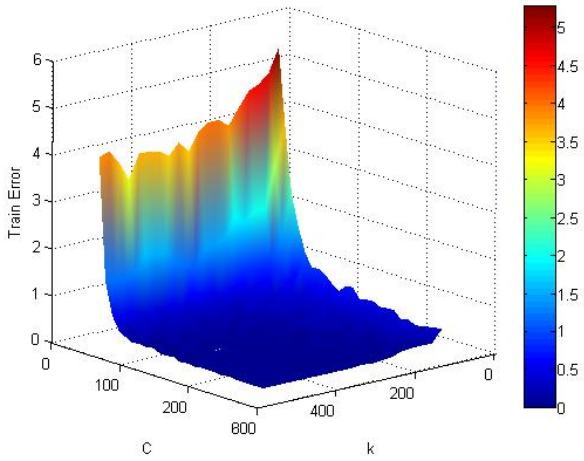

Figure 6: Train error vs SVM regularization parameter (C) and # of principal components (k)

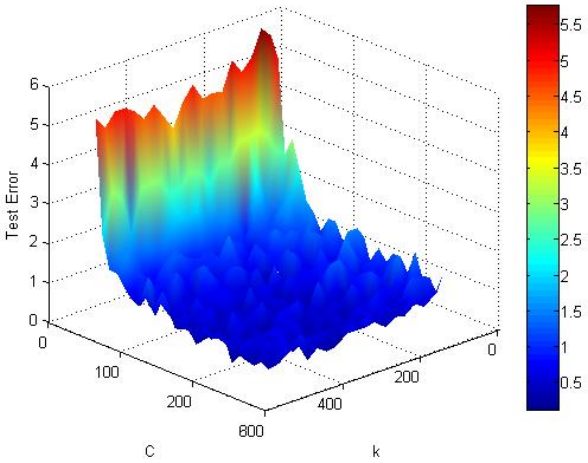

Figure 7: Test error vs SVM regularization parameter (C) and # of principal components (k)

IV. Implementation on the STAIR Robot

The door handle identification and classification algorithms were implemented on the Stanford AI Robot (STAIR 1) for the purposes of making the robot physically open a door. This robot, which has served as a test bed for multiple AI and computer vision projects, is comprised of a Neuronics Robotics Katana 6-DOF robotic arm, a Sony DVI-D100 PTZ camera, a URG Hokuyo laser scanner, and a PC running Windows XP (see Figure 8). The Katana robot arm comes equipped with inverse-kinematic libraries to move the end-effector to a desired location in the arm's reference frame. A user can also query the location of the end-effector as well. The PTZ camera was capable of taking 640-by-480 RGB images, and was mounted on a rail above and behind the robot arm. The laser scanner was also mounted on a rail behind the robot, about 1.6m above the ground, and measured distance from itself to objects in its horizontal plane. The accuracy of this laser scanner was approximately 1cm in the range of distances (0.5m - 3m) at which we were operating. The robot was built on a Segway foundation whose motion was controlled using a Linux PC; however, navigation of the robot itself was beyond the scope of this project. The implementation of the machine learning algorithms on STAIR proceeded in three steps. First, the camera was calibrated and the distances between the camera, robotic arm, and laser scanner were measured. Then, a function was written to convert the (*x*,*y*) pixel location of the door handle into 3-dimensional coordinates in the robotic arm's reference frame. Finally, the robot arm was given commands to move to the door handle's location and open the door. Each of these steps is described below.

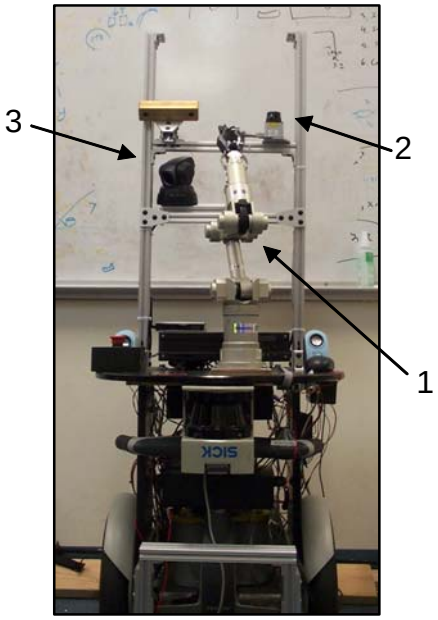

Figure 8: The Stanford AI Robot. Indicated are the robotic arm (1), the PTZ camera (2), and the laser scanner (3).

After photographing 130 pictures of a checkerboard in various orientations, the MATLAB Camera Calibration Toolbox [4] was used to analyze those images and compute the intrinsic parameters of the PTZ camera. With this information, one could convert a pixel in an image into a ray emanating from the camera's origin and passing through the pixel, expressed with respect to the camera's reference frame. By measuring the distances and angles between the camera's and robotic arm's reference frames, we constructed transformation matrices to express this ray in the robot arm's reference frame. (Initially, this was performed with a simple measuring tape, but more complicated refining procedures are described below.) By using data gathered from the laser scanner, the distance from the robot to the door could be determined, and the intersection of this plane with the ray gave the location of the pixel in the arm's reference frame.

In order to verify that the function which converts pixel values into 3D coordinates was predicting reasonable 3D coordinates, a test set was constructed. Each element of the test set was comprised of an image of the robot end-effector and the coordinates of the end-effector in the arm's frame as reported by the arm. Because the laser scanner was mounted too high to record the distance between itself and the robot arm, laser scanner data was "faked" by using the actual coordinates of the arm to calculate what distance the laser scanner would have reported if it had been able to measure that distance. This had the added benefit of removing the laser scanner's 1cm measurement error from the verification procedure.

To get an initial guess for the distances and angles between the camera, laser, and robotic arm reference frames, the distances and angles were measured by hand with a measuring tape and angles were calculated by moving the robot arm along known horizontal paths and querying the arm's location. However, the transformation matrices computed from these estimates produced large (~20cm) error between the predicted points and actual points in the verification test set. By physically positioning the robot arm as close to the camera as possible, we could query the robot arm's position to learn the approximate location of camera's origin with respect to the arm. The same technique was used on the laser, and these new values produced error of ~12cm. In order to reduce the error further, the MATLAB function "fminsearch" was used to search the space of possible translation vectors and rotation angles between the camera and arm to find the values that would minimize the sum of the norms of the error between the predicted and actual points. A regularization term was added to the cost function to penalize the search algorithm for answers that were too far from the measured points. Additionally, we further searched the parameter space of translations and rotations around our guess by writing a MATLAB GUI that would permit the user to click in a series of 2D plots to specify pairs of parameters, and the GUI would plot the predicted points and actual points. This "point and click" method of user-driven parameter tuning produced results that were similar to the results obtained from using MATLAB's optimization function. The errors were driven down to below 3cm in the horizontal directions and 6cm in the vertical direction. Because there was no danger of damaging the arm from error in the vertical direction, we tuned the parameters to reduce the horizontal error at the cost of increasing slightly the vertical error. Figure 9 shows the results of this parameter tuning on the predicted points.

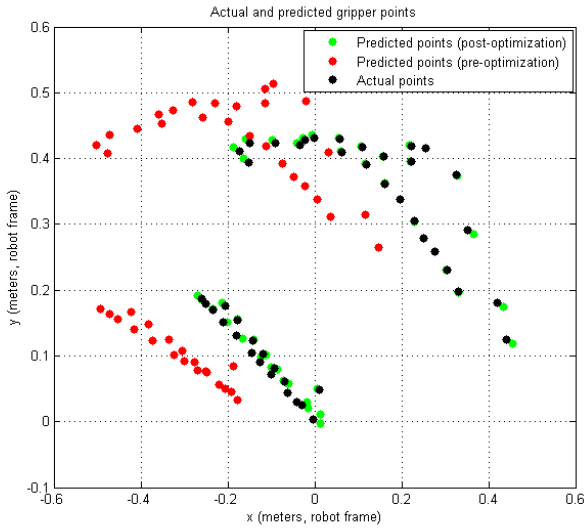

Figure 9: The test data we used to fine-tune the transformation matrix parameters. Our initial measurements of the translations and rotations between the robot frame and camera frame produced transformation matrices which predicted the points shown in red. After tuning the parameters, we predicted the green points, which are much closer to the robot's actual position (black).

Once we had calculated the 3D point of the door handle, the robot arm was driven to a point slightly above the door handle. Then it executed the "door opening action:" it moved its gripper straight down, pushing down the handle of the door. Because the pixel returned by the identification algorithm was shifted to be above the handle portion of the door handle, the gripper would only engage with the door handle itself, and maintain a safe distance from the pivot point.

V. Results

When the entire system was tested with the robot in front of a door, the robot was able to identify the door handle, correctly classify it, find an appropriate point on the door handle to push on, navigate to that point, and execute the door opening action on the door handle. However, the pixel-to-arm-coordinate transformation program required a large amount of parameter-tuning before it was able to position the arm correctly. That is, although the image processing algorithms correctly identified and classified the door handle, the physical implementation leaves something to be desired. Despite having tuned the parameters to achieve acceptable levels of error in the test set of images and gripper positions, we experienced a much higher level of error when testing the robot on an actual door. This suggests that, while the method for tuning the coordinate transformation parameters to minimize the prediction error on the image/gripper point pairs worked well, the test set did not reflect how well the system would perform as a whole. Thus, it would be desirable to construct a new test set of image/gripper point pairs, this time using actual door handles and real laser data. This would permit us to tune the parameters to perform particularly well on the regions of the image known to contain door handles.

VI. Future Work

There are several extensions we hope to add to this project. For the door handle type classification, we would like to add more types of door handles. Preliminary results have been obtained with the addition of a spherical doorknob (for a total of four classes). Using the exhaustive search to find the best values for the SVM regularization parameter and number of principal components, the test error is approximately 1.1% (with training error of ~0.52%). Since the features of spherical doorknobs may tend to look similar to round elevator buttons in a planar image, it may take more clever features to increase the accuracy above 99%.

We also need to do much more testing with the robot on different door handle types, especially types that the identification/classification algorithms were not trained on.

Most of the error in experiment was due to the difficulty of tuning the coordinate transformation parameters. One possible area of future work would be to implement an online machine learning algorithm to learn the parameters. Such an algorithm would automatically move the arm to a known location, take a picture of it, query the laser, and identify the gripper in the image. With a large test set of this nature, one could implement a simple learning algorithm such as weighted least-squares which would map the pixel values and laser distances to the 3D robot arm coordinates. Another option might be to employ a vision feedback system for the robot arm positioning.

Acknowledgments

We'd like to thank the CS department AI lab and Professor Andrew Ng for use of the STAIR robot. We'd also like to thank the members of the AI lab, specifically Ashutosh Saxena, Morgan Quigley, Brad Gulko, and Rangan Srinivasa, for their help in implementing our software on the robot.

References

[1] http://svmlight.joachims.org/

[2] http://svmlight.joachims.org/svm_multiclass.html

[3] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *NIPS 18*, 2006.

[4] http://www.vision.caltech.edu/bouguetj/calib_doc/