CS 4758
Robot Navigation Through Exit Sign Detection

Aaron Sarna Michael Oleske Andrew Hoelscher

Abstract

We designed a set of algorithms that utilize the existing corridor navigation code initially created by Cooper Bills and Joyce Chen [1] to allow an AR Drone Parrot to navigate a building by steering towards exit signs. We trained an SVM on a set of about three hundred representative images, using a feature space defined by an image histogram, as well as a few other features. To improve estimates of the location of the exit sign, we used a Kalman filter, whose dynamics matrix was determined by regression over a set of several hundred images. Finally, to navigate towards the exit sign, we used a P-controller to adjust the yaw of the robot, while keeping the roll and pitch at a constant value.

1 Introduction

While the current implementation of the corridor-following code provides functionality to successfully and robustly follow corridors in an indoor environment, it is limited to going straight and turning corners [1]. In order to demonstrate that such capability can be utilized for advanced navigational problems, we decided to build an application that will navigate an AR Drone Parrot around a building, following exit signs. Since, according to building codes in the United States, exit signs should form a sort of "breadcrumb trail" from any point in the building to an escape point, such an application would allow an autonomous aerial vehicle such as the AR Drone to escape a building from any initial point. We limited ourselves to a single floor, since, while there is a library that will fly an AR Drone up a flight of stairs, there is no implementation for flying down, a non-trivial problem in its own rights.

Once the robot has been placed in the environment, we use the corridor-navigation code to navigate through the corridor until our detector (described below) identifies an exit sign with enough confidence to decide to start flying towards that point in the image. We use a P-controller to turn the robot so that the exit sign is centered in the image. Once we have reached the exit sign (the area of the exit sign in the image is greater than a certain percentage), we decide how to act based on readings from SONAR sensors on the front and sides of the robot. Once we have either turned or determined that no turn is needed, we start to look for a new exit sign to fly towards. Since our exit sign detector will only work if the exit sign is large enough in the image (at a large distance, exit signs tend to be indistinguishable from small red blurs), and the resolution of the forward-mounted camera on the AR Drone leaves something to be desired, this is done heuristically. This can lead to a failure case where the robot will "pace" back and forth between two exit signs.

Using a combination of machine learning and computer vision techniques, we identify the bounding box of the exit sign in the image

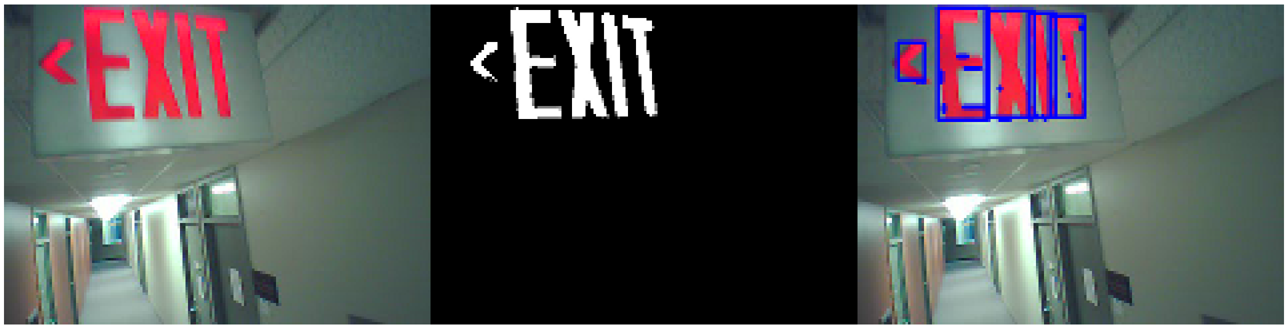

Figure 1: Stages in the image processing. From left to right: initial image, thresholded image, bounding boxes

with a success rate of around 80%. The vast majority of the errors are due to false negatives, which tend to be when the exit sign is very small. This occurs when the robot is very far away from the exit sign, a problem which is handled by the rest of our pipeline. In order to correct for errors in the detection, a Kalman filter is applied to the results, which provides a robust detector.

2 Robot

The robot used was a slightly-modified AR Drone Parrot. In order to support the corridor navigation code and our door and corner turning code, the robot was mounted with three SONAR sensors, forward-, left- and right-facing. These sensors communicated with the computer running the code over an XBee wireless network.

3 Algorithms and Design

3.1 Classifier

To determine where an exit sign is in an image, we first run the image through a pipeline of image processing routines. First, we perform color thresholding on the image. The result is a binary image, where each white pixel represents pixel in the original image whose red component is above a certain value, and whose green and blue are below another value. Black pixels in the binary image are pixels in the original whose values do not meet this criterion. We then find bounding boxes around each of the connected components in the binary image, which represent potential exit signs, or letters of the exit sign (see figure 1). A bounding box is removed if it is too small (the area is less than 20), or if the center of the bounding box is below the middle raster of the image. Since the AR Drone flies below the level of the exit signs in the hallway, all exit sign components will be in the upper portion of the image.

Each of these bounding boxes could be part of an exit sign, or an exit sign in entirety, so we then feed each of these bounding boxes through a merge step before feeding the result to an SVM.

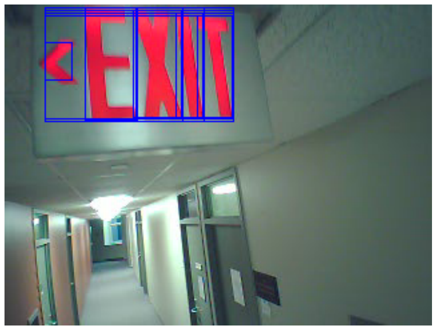

Figure 2: All the bounding boxes whose features are fed into the SVM. There are 15 in this image.

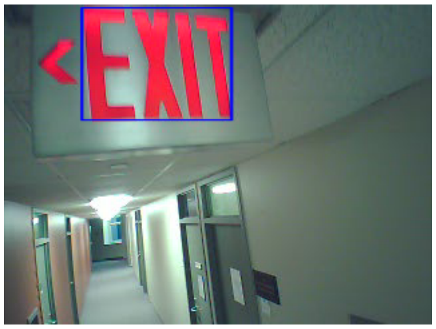

Figure 3: The final bounding box.

The merge step takes two bounding boxes, and will create a new bounding box that encompasses both of them if the original pair have centers that are roughly collinear. This is to prevent having to look at a large number of potential exit signs, and make the corresponding number of classification calls (see figure 2). Additionally, if we have already gotten an initial estimate of the location of the exit sign, we will choose a bounding box with a top-left point closest to the previous location, and only look at merges between this box and the other good boxes.

The merged bounding boxes are then fed through the feature extraction routine, which creates the feature vector for the SVM. The first three values of our feature vector are the x and y values of the top-left corner of the bounding box and the aspect ratio (height / width). The remaining values are a histogram over the colors in the patch. Through repeated use of k-fold cross validation, we determined that the optimal number of bins for this histogram are 15 bins for red, 7 bins for green and 9 bins for blue, giving a feature vector of length $15 * 7 * 9 + 3 = 948$.

At this point, we have a feature vector for each of the merged bounding boxes, which we feed to an SVM [2]. We choose the bounding box with the highest confidence to be our guess as an exit sign (see figure 3). Using this method we are able to get around 80% accuracy.

3.2 Navigation

The robot initially starts in the corridor navigation routines, as described in [1]. It will fly straight through the hallway until the classifier fires on a red blob with high confidence. We then start a Kalman filter to minimize errors in our exit sign detection. The training of the Kalman filter is described in the Experiments section below.

We set the pitch of the robot, which controls forward motion, to a constant value, and vary the yaw, which controls lateral motion, to equal $k_p * (x_{imagecenter} - x_{exitsign})$ where k_p is a proportionality constant. In other words, we use a simple P-controller to align the center of the exit sign with the middle column of the image. Since the robot maintains approximately a constant height, we do not attempt to adjust or center the y-coordinate of the exit sign. We stop moving towards the exit sign as soon as the bounding box of the exit sign takes up a set fraction of the image. At this point, we decided what action to take before beginning to look for another exit sign.

We maintained a state machine to help us decide where to look next based on the SONAR sensors. We first check for open space in front of the robot. If there is no space, we are at a wall and we turn either left or right, whichever way has a bigger reading on sonar. Although this may result in a wrong turn, it allows for a faster decision to be made and another exit sign should be in view that can help the robot get back on track. If there is space in front of the robot, we enter a state called "looking for turn." It is most likely that there is a doorway or a hallway to turn down when this occurs, so the sonar sensors are used to detect a sudden increase in depth. A sudden increase in depth will occur because an open door or a perpendicular corridor will have deeper distance for the robot to read then the closer wall. Readings are passed through a mean filter to compensate for noisy sonar sensors. The search for side depth increases and front depth decreases is done for a fixed number of iterations to facilitate the case where the exit sign is indicating that we should pass straight through an open door in front of the robot, at which point will go back to looking for a new exit sign. After the robot detects a sudden side depth increase, the robot is told to turn in this direction by setting the yaw. This is the turn left or turn right state. It will complete turn by using readings from the front sonar sensor. The front sonar sensor will originally be reading a long distance as it is pointed down the corridor. As it turns, the distance will get shorter and when it hits a threshold we denote that the robot is in state two of turning. To complete the turn, the front sensor looks for a long distance down the new corridor (which is through the doorway). Once this is detected, it ends that state by setting the pitch to go forward and we start the procedure of looking for an exit sign again.

However, due to a lack of functioning SONAR sensors, we later had to transition to an inferior version of this. Since we only have two working SONAR sensors, and there are none available for sale in Ithaca or the surrounding region, we now use the built-in rotation sensor to complete our turns. This means that we cannot decide to move forward if there is a large open space, since we have no readings from the front sensor. Since we decided that the two side sensors are more important than the front sensor, our new algorithm works as follows. When we reach an exit sign, we decide to turn towards the side with the most open space. We then turn 90° based on the internal rotation sensor. We tried to decide to move forward if the two side readings were both less than a certain value, but this proved to be error-prone, due to the noise of the SONAR sensors. Once we have completed this turn, we proceed as before.

4 Experiments

4.1 Data

We have two main data sets, one used for training the SVM, and the other used for determining the transition matrix for the Kalman filter.

The first set consisted of one hundred and fifty images of exit signs, each with an accompanying text file describing the bounding box of the sign in the image, as well as another one hundred and fifty images without exit signs, including images of hallways, posters and other signs. This was acquired by flying the robot manually, and saving images from the robot's front image stream. It includes four buildings on campus with multiple floors on each building, and around thirty distinct exit signs, each from multiple angles and distances.

The second set is images from ten autonomous flights towards exit signs with fifty images each. Different distances and yaws were used for the various flights and images. We hand labeled the top-left corner of the exit sign and its area in each image. We ran a linear regression to determine the transition matrix for the expected Δx and Δy based on the previous x and y, the square root of previous area, and the yaw. We set the covariance matrices through educated guesses, assuming that the camera was very accurate and the prediction would not be, just given the variability of the flight and the relatively poor fit of the linear regression.

We are happy to contribute our labeled datasets to publicly available databases, but we do not know where would be an appropriate place.

4.2 Off-line Experiments

Our classifier has three parameters whose values are difficult to set, namely, the number of bins for red, blue and green colors in the image histogram. We know that we wanted to keep the number of bins somewhere between 500 and 1500, since this is roughly a good range for SVM feature vectors, and, since exit signs are predominately red, the number of red bins should be slightly higher than the number of green and blue.

In order to determine the best values for these parameters, we ran a series of three-fold cross validation over the SVM training image set described above, changing the number of bins for each time. We ranged from having 13 red bins, 7 green bins, and 7 blue bins to 15, 10 and 9 red, green and blue bins, respectively. We chose these numbers due to some informal testing done prior. The error rates of each of the different bin combinations are shown in figure 4 below.

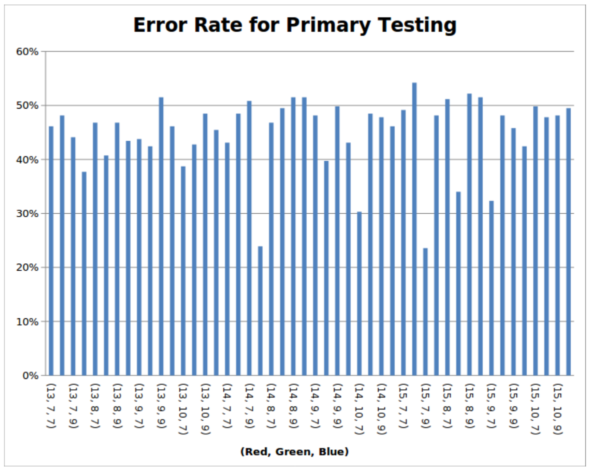

Figure 4: The error rates for the 3-fold cross validation.

We then chose the five bins combinations with the least error, and ran each through a ten-fold cross validation routine. The bin combinations with the lowest error rates were (14, 7, 10), (14, 10, 7), (15, 7, 9), (15, 8, 8) and (15, 9, 7). The final error rates for each of these are shown in figure 5. Since (15, 7, 9) had the lowest error rate on the final test, this is the bin counts used in the final classifier.

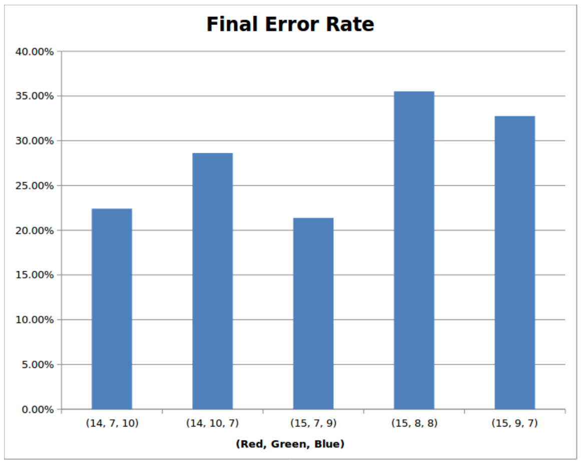

Figure 5: The error rates for the 10-fold cross validation.

4.3 Robot experiments

With the classifier and Kalman filter trained we ran a tremendous number of test flights down many different corridors, towards many different exit signs, which indicated to many different types of actions, such as ”go through this door,” or ”turn down this hallway,” or ”continue straight ahead.” With every success and failure we tweaked settings and algorithms manually and gradually improved. Some settings were far too drafty for the robot to be able to handle, so we gave up on those. Furthermore, most exit signs are at the end of corridors, meaning the corridor navigation code tended to enter the “Unknown” environment before the exit sign was clearly distinguishable in the image, meaning that, in these cases, the robot could not even get close to the exit sign. We had a lot of trouble finding spots to work where we would not be disturbing people and were kicked out of a bunch of locations. We also had to deal with extremely flaky hardware that wasted a lot of our time. The sonar sensors repeatedly died on us, the robot wireless cards would not connect, two of the motors on the robot died, and the camera mount cracked when the robot crashed.

At this point, most of the experiments have been relatively successful, though, due to the greedy approach that we take to determining the next action to take, we often choose the wrong way to go, sometimes "pacing" back and forth between two exit signs. With the modified approach to the corner turning algorithm, we often choose to turn at an incorrect time, since we have no forward sensor to decide if we can move forward. See the following figures for images of the robot in action.

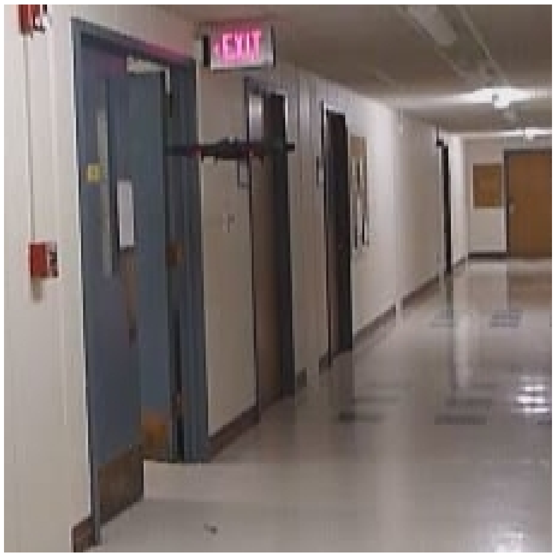

Figure 6: The robot about to turn and fly through a door.

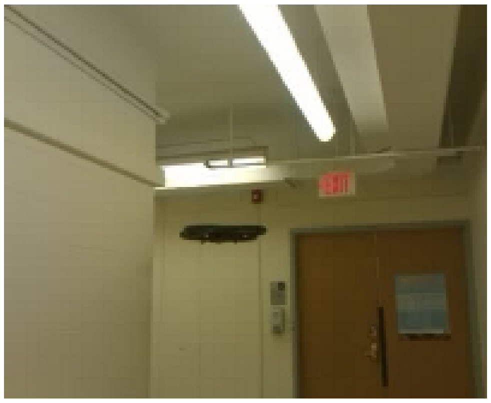

Figure 7: The robot turning around a hallway

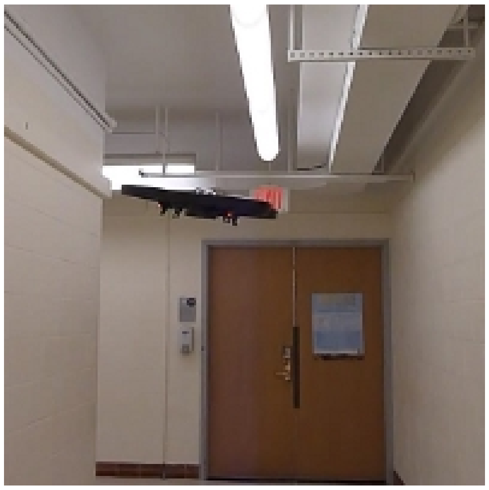
Figure 8: Flying towards an exit sign

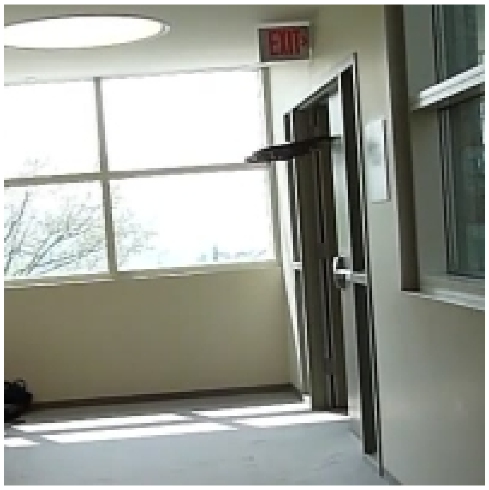

Figure 9: Turning through another door

5 Conclusion

We presented a set of algorithms for navigating a building by following exit signs. We used a basic image-processing pipeline to extract features from patches of red in the image and classify them using an SVM. The output of the SVM is smoothed using a Kalman filter to account for errors. The robot then steers, using a P-controller, to the exit sign, at which point it searches for the next exit sign in the chain. We validated our classifier using 10-fold cross validation, and the navigation through many robot experiments. Though the robot had problems with malfunctioning sensors and other hardware issues, we have shown that capabilities of flying through corridors can be utilized for more advance indoor navigational problems.

References

[1] *Autonomous MAV Flight in Indoor Environments using Single Image Perspective Cues*, Cooper Bills, Joyce Chen, Ashutosh Saxena. To appear in International Conference on Robotics and Automation (ICRA 2011), 2011

[2] T. Joachims, *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.