Autonomous Interpretation of Elevator Panels for Robot Navigation

Mark Baybutt, Blake Carpenter, and Kristen Lurie

***Abstract*— The objective of the work presented in this paper is to develop a method to interpret interior elevator panels for robot navigation. We outline a three-level hierarchical approach, which applies principles from both supervised and unsupervised machine learning techniques. In particular, we will show the development of an algorithm that accepts an elevator panel image paired with a desired floor location and determines the region in the image corresponding to the floor's button and optimal location for a robotic arm to press.**

I. Introduction

WHILE robots exist which can autonomously navigate unknown buildings, one outstanding robotic navigation problem is interaction between robot and elevator. Developing a solution to this challenge would enable robots to traverse previously unattainable floors of a building. For a robot to successfully interact with and navigate an elevator, there are many tasks that need to be performed. These include identification and action upon the exterior panel, entrance into the elevator, detection and alignment with the interior elevator panel, identification and action upon the interior panel button, and exit from the elevator. Klingbeil *et al* [1] touched upon the first area - detecting and manipulating exterior elevator button panels - as an extension of work related to detecting and manipulating door handles. Despite the similarities to exterior elevator panel detection, interior elevator panels are more complex as they contain numerous floor options in various configurations.

The discussion of work performed herein addresses the challenge of interpreting and taking action upon interior elevator panels. In particular, the goal of the project is to develop an algorithm that will identify a location for a robot to press based on the desired floor location with ultimate implementation on the STAIR.

II. Overview

Our proposed approach divides the main task into three simplified sequential steps consisting of:

1. Identifying buttons and labels (Figure 1A)
2. Interpreting labels (Figure 1B)
3. Assigning text to buttons (Figure 1C)

This procedure is intended to be general as to best interpret any elevator panel.

The procedure starts by identifying an elevator panel's buttons (any object which performs a function when pressed, red regions in Figure 1A) and labels (any character or picture which indicates a button's function, yellow regions in Figure 1A). From these identifications, interpretation of the characters and text on the labels is performed (yellow regions in Figure 1B). Then, based on the position of the buttons and labels, the two can be correlated such that each button has some descriptor of its function. From these three steps, a depression point for each button and a descriptor of these points can be determined. With this information, the robot can ascertain where to press based on the desired floor location.

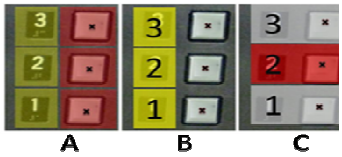

Figure 1: Steps Involved in Interpreting an Elevator Panel

III. Approach

A. Data Collection and Segmentation

Elevator panels vary greatly in terms of button design, orientation and labeling; thus it is extremely important to have not only a sizable training set, but also a training set that captures the many different variances in panel style. To create a training set that reproduced elevator panel variance, we collected approximately 300 images of interior elevator panels.

To aid in cropping positive and negative training images, a MATLAB application was written which enables users to select regions of the image and associate with desired tags. Figure 2 shows a portion of an image with the cropping and tagging scheme implemented. Four tags have been defined to identify regions of images: *labels* (red boxes), *labeled-buttons* (yellow boxes), *unlabeled-buttons* (green boxes), and *negative* (blue boxes). After the user crops and tags the image appropriately, the application automatically selects varying sized portions of the unselected regions of the image (i.e. non-buttons or non-labels) to include in the *negative* training set. To provide added flexibility, the application also provides the ability to specify regions of the image to explicitly include in the *negative* training set (such as key holes which may look similar to buttons).

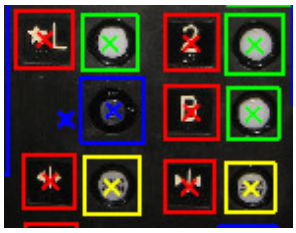

Figure 2: User Controlled Image Tagging

User specified information is stored from tagging including: type of tag (label, labeled-button, or unlabeled-button), description for labels and labeled-buttons (e.g. alarm, floor 1, etc.), coordinates and dimensions of each of the tagged regions, and the button, which each label describes. These

Submitted as a project Final Report for Stanford University CS229 Autumn 2008.

metrics are later used as ground truth (GT) evidence to evaluate the performance of each stage of the algorithm in addition to overall performance.

B. Button and Label Identification

A sliding-window object detector (SWOD) [2] was utilized to identify regions within test images, which contain buttons and labels. We utilized 80% of the tagged images for SWOD training to create two image models, a Label model (using both *label* and *labeled-button* images in the positive training set and *unlabeled-button* and arbitrary background images in the negative training set) and Button model (using *unlabeled-button* and *labeled-button* images in the positive training set and *label* and arbitrary background images in the negative training set). The remaining 20% of tagged images were used for testing.

To quantify detection results, the following metrics were defined: a true positive (TP) detection indicates the detection box centroid fell within the GT detection box, a false positive (FP) detection either did not classify the item correctly or a TP detection with higher probability already identified the GT item in the image.

The two models were applied to approximately 60 test images to gauge overall performance. The output of the SWOD for each image was a listing of detections' size, x- and y-coordinates (in pixels), and the probability in which it matched the applied model. The resulting detections had 100% recall, but incredibly low precision as compared to GT. The intuition gained from these initial findings was the models generated a considerable number of detections per image, some of which correctly identified the item of interest, contributing to high total recall; yet the vast majority either did not accurately identify the item or was a repeat detection of the item, contributing to low precision. Refinement to the training sets were made and new models generated and retested; little improvement was seen and from these results, we concluded that an intelligent method for downselecting detections was required to increase precision at this stage of the process and improve accuracy of the overall approach.

To accomplish the downselect of detections, a three-phased methodology was implemented. A static probability threshold of 0.6 was initially implemented where detections above 0.6 reported probability were kept while the rest were discarded. This approach worked reasonably well in most test cases; however, failed in instances where detections did not return relatively high probability. To counteract this effect and provide a means to generate a more natural threshold per detection set, a dynamic probability thresholding technique was implemented wherein the maximum and standard deviation (STD) in probabilities were calculated based upon each set of detection. Probabilities (pr) which fell within the set of $[\max(\mathrm{pr}), \max(pr) - A * std(pr)]$ were retained, while detections with probabilities outside of this range were discarded. The coefficient '*A*' was a control parameter that was experimentally determined through the use of precision-recall (PR) curves to find an optimal solution (see Figure 3 and subsequent discussion). The reduction from raw to thresholded detections can be seen by comparing Figure 4A and Figure 4B.

The second step in the downselect of detections was a pruning phase. This phase was motivated by the fact that in many practical applications range finding capabilities could assist in determining the distance from the camera to the elevator panel (such is the case with STAIR). From this distance, a calculation of expected button/label detection size in pixels can be determined as the physical size of buttons/labels adheres to guidelines set forth by the American Disabilities Act (ADA) [3]. In the absence of such distance information, the ratio of GT box sizes (in pixels^2) to source image size (in pixels^2) for all training images was calculated as it was concluded every training and test image was taken from the same range of distances (3-4 feet) from the elevator panel. The values satisfying Equation 1 and Equation 2 were used for $\mathrm{Prune}_{\min}$ and $\mathrm{Prune}_{\max}$, respectfully.

$$\mathrm{Prune}_{\min} = \max\begin{bmatrix} \min\left(\frac{\mathrm{GT}_{\mathrm{area}}}{\mathrm{Image}_{\mathrm{area}}}\right), \\ \mathrm{mean}\left(\frac{\mathrm{GT}_{\mathrm{area}}}{\mathrm{Image}_{\mathrm{area}}}\right) - 2 * \mathrm{std}\left(\frac{\mathrm{GT}_{\mathrm{area}}}{\mathrm{Image}_{\mathrm{area}}}\right) \end{bmatrix}$$

Equation 1

$$\mathrm{Prune}_{\max} = \min\begin{bmatrix} \max\left(\frac{\mathrm{GT}_{\mathrm{area}}}{\mathrm{Image}_{\mathrm{area}}}\right), \\ \mathrm{mean}\left(\frac{\mathrm{GT}_{\mathrm{area}}}{\mathrm{Image}_{\mathrm{area}}}\right) + 2 * \mathrm{std}\left(\frac{\mathrm{GT}_{\mathrm{area}}}{\mathrm{Image}_{\mathrm{area}}}\right) \end{bmatrix}$$

Equation 2

Satisfying these equations ensured the pruning values will not exceed the bounding limits of the data set (i.e. maximum and minimum values). For each test image, the pruning values were multiplied by the image size to determine the expected range of detection window size. Detections which fall outside of this range are discarded. This downselect step is most evident in the upper third of Figure 4B and Figure 4C in the removal of the relatively small button detections (seen in red) and the relatively large label detections (seen in yellow).

The final step of downselecting detections is to remove overlapping detections. A variety of methods were applied to achieve this including, probability weighted k-means, probability weighted k-median, and probability dominated percent overlap removal. Results from weighted k-means caused resulting centroid positions to be drawn away from correctly identified button/label regions when detections with high probability were clustered together. A range of 'k' values were experimented with; however, the effect persisted in a number of trials. To eliminate the detrimental averaging effect of k-means, a weighted k-median approach was applied. This approach worked reasonably well in assigning the k-centroid to the detection with highest probability in the assignment cluster; however, it was found to be computationally expensive and the results from probability dominated percent

overlap removal performed equally as well at less computational expense. Since the ultimate goal of this algorithm is to be implemented in near real-time, the percent overlap removal method was deemed preferable. In this method, detections are compared to one another and if an overlap percentage above a control parameter '*overlap*' is exceeded, the detection with higher probability is retained while the remaining detections associated with the overlap region are removed. The '*overlap*' parameter was experimentally determined through the use of PR curves to find an optimal solution (see Figure 3).

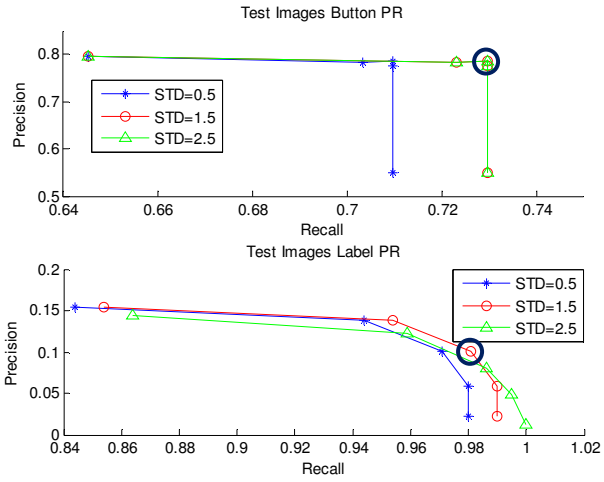

Figure 3: Button (top) and Label (bottom) Precision-Recall Curves

The PR curves seen in Figure 3 assisted in determining optimal values for the STD coefficient '*A*' in the dynamic thresholding step and percentage '*overlap*' in the percent overlap removal step. Each curve in the plot represents a variation in STD while each data point represents a variation in percent overlap. A sweep of STD values from 0.5 to 3.0 and overlap values of 0.1 to 0.9 were made on all test images. A subset of these results is shown in Figure 3 for clarity. The PR curve shows interesting behavior for buttons as increasing STD from 1.5 to 2.5 does not increase precision or recall. The intuition from this observation is that all button detections are within 1.5 standard deviations from the maximum probability. Nevertheless, increasing the percent overlap threshold to an optimal point does improve recall while not sacrificing precision. The optimal corner point circled in dark blue represents 60% overlap (i.e. detections exceeding 60% shared overlap are trimmed to one detection). Increasing this overlap threshold causes precision to plummet as a greater number of detections are retained that identify the same GT location.

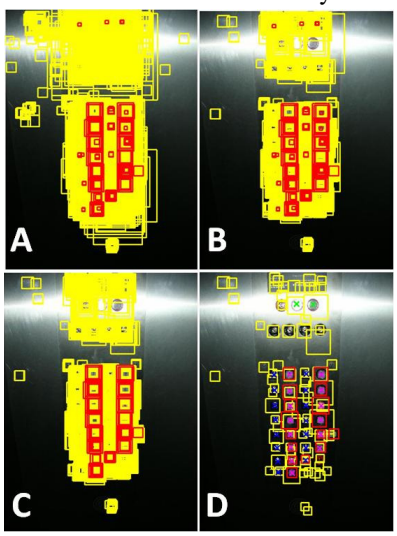

Figure 4: Visualization of Button and Label Identification (labels in yellow, buttons, in red), (a) Raw SWOD detections, (b) Post-probability thresholding, (c) Post-pruning, and (d) Post-overlap detection and removal

The PR curve associated with label detection behaves in a more typical manner where including a greater number of detections (i.e. increasing STD) increases recall while reducing precision. An optimal combination of precision and recall can be achieved with an STD=1.5 and 30% overlap which jointly maximizes precision and recall. The results from applying this final step in detection downselect can be seen by comparing Figure 4C and Figure 4D.

C. Text and Character Recognition on Buttons and Labels

After labels have been identified by the SWOD, we use character recognition to classify the segmented labels. Three approaches for label recognition were considered: text/character SWOD, support vector machines (SVM), and Tesseract (Google's OCR software). Each approach was tested and evaluated and based upon performance and success rates, Tesseract were selected to implement text and character recognition on buttons and labels.

The text/character SWOD approach aimed to detect characters and image patterns in a label using the SWOD. We created SWOD models for alpha-numeric characters as well as common elevator characters, such as ">|<", the door close sign. From each label, we segmented individual characters as positive training samples. For example, label "23" was segmented into a "2" and a "3". The "2" was used as one positive training sample for the "2" model with the "3" and other parts of the label as negative training samples. Each model was run on a single label. The resulting classification detections from each model were analyzed and based on these detections a joint classification was created. A "23" label, for example, ideally has many detection boxes from the "2" and "3" models, while every other model has significantly fewer detections. Based on the position of the "2" model detections relative to the "3" model detections, it can determined that the label is a "23". This approach did not work well in practice as both precision and recall values were low and no more data was available.

The second approach attempted to use SVM on a label to determine its type. The feature vector was the intensity of each pixel when reduced to a 50x50 pixel black and white image. A model was created for numbers 1, 2, and 3. While the training set error for each of these models was 0%, there was no separation between positive and negative examples and thus error was high. The models were clearly overfitting the data. We decided not to pursue SVM any further as improving as it would require us to accurate segmentation each digit from a label, which proved complicated.

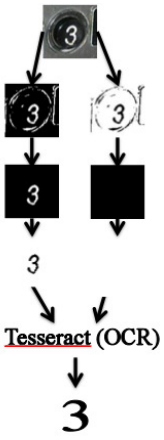

Figure 6: Tesseract Process Path

The third approach uses Tesseract to classify a binary, processed version of a label. Tesseract recognizes black pixels on a white background and tries to classify the pixels as characters in its own dictionary. After experimenting with Tesseract, we learned that it would return text strings that do not correspond to floor labels, if given large areas of black pixels that did not resemble anything or a readable character with noise surrounding it. Also, in some cases it would incorrectly classify an image that clearly resembled a known character.

Thus, images of labels must be converted to noiseless binary images with only the characters remaining to be properly classified. These images are created by converting the image to binary using a dynamically selected threshold based on image contrast and then removing regions of noise (i.e. long, thin lines, regions of a few pixels). Due to the fact, it is difficult to capture a noiseless digit with our image processing techniques, we try to isolate characters using eight different images with small variation in threshold and region detection. Ideally, Tesseract then classifies the unclear images as non-floor text strings and the clear images as the true value. The results are then analyzed and the best classification is returned as the final classification for the label, as seen in Figure 5.

Because Tesseract can only classify digits and text characters, it returns non-floor text strings for labels with pictures or many characters, such as the bell label or a "CLOSE DOOR" label. Labels that are classified as non-floor text strings are removed from future steps. This result is sufficient as the robot does not need to push non-floor buttons for navigation. The output of the label text identification step is shown in Figure 6.

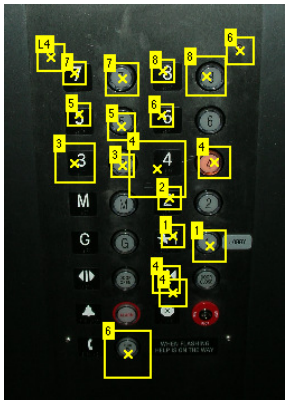

Figure 7: Labels classified through Tesseract

In practice, problems arose that ultimately led to poor accuracy for this approach. Figure 7 shows confusion matrices for Tesseract versus ground truth label classifications. Results show that there is a significant amount of error in this process. Error comes from both the image processing step, as it is difficult to eliminate noise and retain the characters, and the OCR step, which often misclassifies images which seem clear to the observer.

D. Classification of Buttons and Correlation with Labels

After recognizing the label text, we then assign text to buttons with a three step process. This means we will assign valid floor text to the floor buttons and no text to non-floor

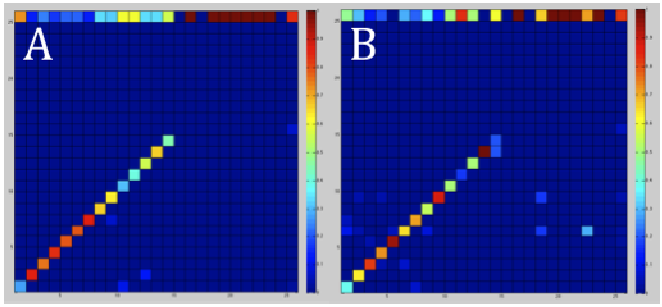

Figure 5 Confusion Matrix.(A is labels, B is labeled buttons) with y-axis is ground truth label type and x-axis is OCR classification. Higher probabilities correspond to more red shading. The diagonal portion of the matrix contains number values, the last row and column contain non-floor text strings, and the other values are letters. Numbers are occasionally and letters are frequently classified as gibberish. Labeled-buttons are classified less accurately.

buttons and FP buttons. First, we used SVM with a linear kernel to train a model, which correlates labels to buttons [4]. Five features are defined for each button and label pair: distance and percent overlap between button and label, overlap between button and other labels, and prediction on whether label lies on the correct side of the button, which is determined by the ratio of buttons to labels on the left and right of the button. This model assumes that a button and its related label have roughly the same y-coordinate, but does not assume a placement for the label, which allows us to capture labels that are to the right, left, and on top of a button as well as multiple labels for a button. The precision and recall are 98.76%/98.19% and 99.32%/99.24% for the testing and training sets, respectively. (Figure 5A)

After buttons and labels are correlated, the label text is assigned to the button in which it is paired. If there are multiple labels for a single button, we choose the label text over the label-button text, because Tesseract more accurately classifies labels. Precision and recall for this step are 99.90%/99.71% and 99.88%/99.92% for testing and training sets, respectively. Precision and recall both increase, because many errors from correlation do not contribute to properly identifying button text (*i.e.* a button with three identical labels are matched to only two of them), shown in Figure 5B. This process eliminates some false positive buttons, since buttons that are not assigned to labels are likely false positive detections.

We applied the SVM models in two ways: by including all labels and by only including valid floor labels. Although accuracy of each method was approximately the same, we chose the latter because as it is impossible to assign valid button text from a floor from button text to valid floor label that are not associated with valid floor labels. However, by eliminating invalid floor labels it is possible to incorrectly correlate buttons with valid floor labels as the SVM model relies on the labels that are located in the same row as a

button. With higher accuracy in the label identification step and the same high recall and low precision for the label model, applying the SVM with only valid floor labels makes the most sense as we would correctly eliminate false positive labels before running the SVM.

Because there is poor accuracy identifying label text, we attempted adjust some of the misclassifications, such that all floor buttons are assigned to valid text and all non-floor buttons are assigned to no text. The buttons are fit to a grid by using the detected center point of each button. (Figure 5C) Then, the pattern of numbers on the panel is determined by identifying the most common difference between floor numbers across a column and down a row. We use these two numbers along with numeric label detections to generate the ideal arrangement of numbers. This process oversimplifies the typical elevator panel by assuming, for example, that floor labels are only numeric and always vary by the same number across a row and a column. However, this model of the elevator panel does particularly well when only a few floor buttons are improperly labeled. (Figure 5D).

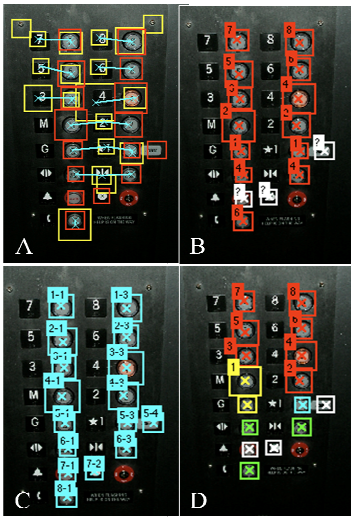

Figure 8: Assignment of Text to Buttons: (A) Buttons (red) are correlated with their labels (yellow) as shown with blue lines; (B) Buttons absorb the text of the labels to which they were matched. Labels that represent floors are in red, the rest are in white; (C) Button locations are placed in a grid; (D) Labels are adjusted based on numeric labels. Red and white boxes represents no change in text, green is improvement in classification, and blue is improper adjustment.

IV. Results

In addition to the evaluation of individual systems, overall system performance was evaluated. For each of the three main steps in the process, we started with ground truth data and carried the process to completion to evaluate the efficiency of each step; the results of these tests are presented in Table 1.

There is no significant difference between system performance with and without adjustment of button text. However, recall for for the last step decreases signficantly (99.85% to 68.85%) when we add the adjustment step confirming that many of the assumptions in the adjustment step are not valid. In general the majority of the error can be attributed first two steps. By improving the accuracy of these steps, we will likely eliminate the need for the adjustment step.

Table 1: Component-wise system test results

	No Adjustment		With Adjustment	
Component	**Precision**	**Recall**	**Precision**	**Recall**
Overall System	***36.11%***	***30.34%***	***35.14%***	***29.08%***
Button and Label Identification	36.11%	30.34%	35.14%	29.08%
Recognition of Label Text	100%	59.24%	100%	61.72%
Assignment of Text to Button	100%	99.85%	100%	68.85%

V. Discussion and Future Work

The results from the proposed approach show promise for an autonomous method for interpretation of elevator panels. Despite arguably low precision and recall for overall system performance, the approach applied a variety of methodologies and machine learning principles in search of an optimal and robust solution. From these varied approaches for each stage, the best option was selected based upon standard performance metrics. A number of refinements were identified and are currently underway. To improve the overall accuracy of button and label identification, fine-tuning of the label model would more accurately classify true positives and improve initial precision. The initial mentality of desirable high label recall assumed the text and character recognition would accurately identify TP labels while rejecting FP as unclassifiable. In practice, this was an improper assumption as the character recognition step is often able to identify text from FP labels. To improve character recognition, we seek to develop a process to segment individual characters from the labels and then use an SVM to classify individual characters and ultimately label text strings. Finally, button and label correlation can be improved by identifying features that better suit typical output from the label text recognition stage.

References

[1] E. Klingbeil, A. Saxena, and A. Y. Ng, “Learning to open new doors,” AAAI 17th Annual Robot Workshop and Exhibition, 2008.

[2] Stephen Gould, Andrew Y. Ng and Daphne Koller, “The STAIR Vision Library,” http://ai.stanford.edu/~sgould/svl, 2008.

[3] “ADA Evaluation,” ThyssenKrupp Elevator American Business Unit, http://www.thyssenkruppelevator.com/images/brochure_images/pdf/06%20ADAEvaluation_11_06.pdf, 2006

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