Janitor Bot - Detecting Light Switches

Jiaqi Guo, Haizi Yu

December 10, 2010

1. Introduction

The demand for janitorial robots has gone up with the rising affluence and increasingly busy lifestyles of people [1], both in offices and personal homes. All too often people forget to switch off lights, close doors, throw away garbage and so on; this is where a janitorial robot would come in handy. This kind of robot must have the necessary characteristics of being able to autonomously navigate through its surroundings, and also detect and recognize objects of interest for its assigned tasks. In this project, we focus on the latter problem for the specific case of turning off lights. The robot needs to know where the light switch is in order to turn the lights off, so the visual system must be able to precisely detect the 2D location of the light switch on the wall.

In computer vision, there have been numerous object detection methods [2, 3] that involve extracting Haar features and forming integral images from the original images, before passing them to a classifier that is trained using various machine learning algorithms. These have achieved high accuracy and low false positive rates for the detection of faces, as well as other objects, but necessary modifications are required for detecting objects related to janitorial tasks due to the difference in characteristics that we could make use of to improve the efficiency of detection[4, 5]. We aim to design an algorithm that detects a light switch as efficiently as possible.

Switches come in all kinds of sizes, and the range of lighting conditions, the fluctuation in angles that the image of the switch was taken from, and the decoration on walls as well as switches make the learning problem difficult. But a common characteristic is the rectangular switch plate shape, which we will exploit in tandem with a classifier, as described in this paper. For the purposes of this project, we are assuming that the robot is able to look at the wall from a one-yard distance, at a height range of roughly half a meter above and below the normal height of a light switch.

2. Data Collection and Preparation

The training examples for machine learning were obtained by taking photographs of interior walls in various buildings around campus, at a distance of roughly a yard away from the wall and a height range of roughly 1 meter around the normal height of a light switch, since the robot will also be viewing the wall at this distance and height. They were taken directly facing to the wall or at a slight angle from the perpendicular. These full photographs were resized to 640x480 pixels, which is the resolution of the image that the robot would see.

Light switch plates usually have a height-width ratio of 1.6:1, so we chose our positive training examples to be 160x100-pixel images cropped from the resized full photographs, with individual switches roughly occupying 130x80 pixels and parts of double or triple switches occupying 130x90 or 130x100 pixels (see Figure 1a). Negative training examples were 160x100-pixel images of objects, parts of objects, or wall surfaces located at roughly the same height as a light switch (see Figure 1b). These are considerably abundant compared to the positive examples. An image that contains part of a switch is considered a negative example.

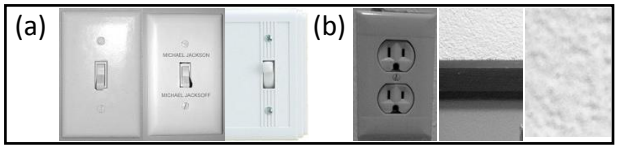

Figure 1 (a), (b): Instances of positive and negative training examples respectively. Note that in (a), half of a double switch (third example) is taken as a training example.

The images were passed through a selective Gaussian filter to remove some noise, and used to train the classifier as grayscale images. We chose to use grayscale images to reduce the number of raw features that we are starting with, as color information might cause unnecessary overfitting.

After training the classifier and testing on a full image of a wall (not used for creating the original training set), some of the false positives detected were placed in the training set to retrain the classifier to increase its accuracy. Still, due to the lack of a sufficiently large image database for training the classifier as of yet, we designed our detection algorithm to reduce the chances of the SVM detecting false positives by making use of the common characteristics of light switches.

3. Method

3.1 Testing Training Algorithms

We first trained both an artificial neural networks (ANN) classifier and a support vector machine (SVM) classifier using 16000 raw pixel values as features. We then trained these two classifiers by performing PCA on the pixel values of the cropped images, looping through the number of principal components retained to find the optimum number to keep that would give us the lowest error rates. The error rates (see Table 1) suggested that we should use PCA with 81 components kept as features, and SVM as a classifier for the detection of the switch in the full image.

	Raw, ANN	Raw, SVM	PCA, ANN	PCA, SVM
Training Error	3%	0%	2%	0%
Test Error	10%	12%	9%	5%

Table 1 Error rates for each classifier. Raw = raw pixel features used. PCA = PCA with 81 components kept as features. ANN = ANN classifier used. SVM = SVM classifier used.

These errors are given for the original training set that we started out with, without adding the false positives that were added later from our experiments.

3.2 Sliding Window

To detect the 2D location of a switch, we implemented a 160x100-pixel sliding window that moves over the whole image in a zig-zag pattern and determines if the area that it bounds contains a switch using the detection algorithm described in the next section. To improve the efficiency of the search, the movement of the sliding window is sped up if certain conditions are met, as described later. The sliding window was also used to capture false positive images for retraining the classifier.

3.3 Detection Algorithm

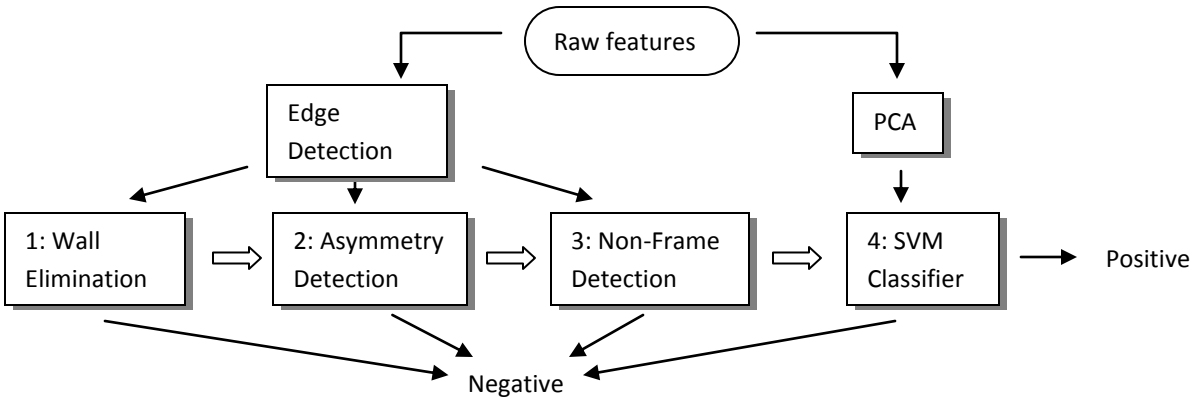

Figure 2 Flowchart for Detection algorithm. Stages 1-3 consitute preprocess detection, which throw out obvious negative image patches based on whether they have the common characteristics a switch has. Image patches surviving these stages are then sent to the classifier to be identified as a switch or not.

In preprocess detection (stages 1-3), we extract edge features from using the Canny edge extraction algorithm, to generate a Boolean valued image (called edge-image), and perform various computations to sift out obvious negative examples such as walls and areas with absolutely no switch-like

characteristics, before we pass it into an SVM classifier (stage 4). This reduces the possibility of the classifier identifying false positives.

Although we are using edge features in all of the first three stages, the features extracted are different in each, as we tuned the sensitivities for edge detection for each stage differently to filter out the right amount of noise and getting the edge features that are useful for each different stage. The sensitivities were chosen as fixed values based on similar computations with examples from the training set, and optimized as best as possible toward the goal of each stage.

3.3.1 Stage 1: Wall Elimination

The majority of non-switch areas appearing in a natural scene are simple wall or wall-like areas with negligibly few features of high frequency. Such an area would have a very small variation in luminance. Thus its edge-image should have a very small number of ones. We set a threshold for the sum of ones in a 160x100-pixel image, below which we classify the image as a non-switch, or an area with too few features to be a switch.

3.3.2 Stage 2: Obvious Asymmetry detection

Light switches are usually situated in the center of a rectangular switch plate, so an individual light switch centered properly forms an image of high symmetry. Upon running some tests, we found that a switch that is part of double or triple switches is still more symmetric than a large quantity of image patches that the sliding window will encounter, which contain incomplete parts of a certain object (including parts of the switches). Image patches like these that are clearly non-symmetric are what we aim to remove from the full image using this asymmetry detector proposed here. We can compute the total number of ones in the upper and lower halves, and left and right halves of the edge-image, and represent the symmetry of the image by the variance of these four numbers. We then eliminated the images with a variance above a threshold chosen to keep image patches of individual light switches or parts of multiple switches.

3.3.3 Stage 3: Obvious non-Frame detection

We also note that a complete light switch would have a perfect or near-perfect rectangular frame in the edge-image, which non-switch areas would not have. Parts of multiple switches also have more of a frame than most non-switch areas. We represent the probability that the image patch has a frame by the sum of the number of ones in a margin around the sides of the edge-image, and then eliminate the areas which have a probability lower than a threshold chosen to retain switches based on values computed from our training set. We chose the size of the margin such that the inner edge of the margin always lies within the switch plate of an individual switch. Images of objects that have frames or partial frames in such area should exhibit large probabilities. This detector could eliminate a switch with a frame that is smaller than the inner edge of the border, but rescaling the image and repeating the search procedure over the larger image can easily solve this problem.

3.3.4 Stage 4: Supervised learning via an SVM classifier

In our experiments, we trained the SVM with a linear kernel beforehand and load the support vectors to make a prediction on whether the image patch contained in the sliding window that had survived the previous three stages is indeed a switch. The SVM was trained using parameters that were automatically optimized for the training set in the training process, such that the k-fold cross-validation estimate of the test set error is minimized.

3.4 Sliding window shifting strategy

In order to speed up the detection procedure, since it would be unfeasible to have a robot stand in front of a wall taking five minutes to make a decision on where the switch is, we slightly modified the normal sliding window movement that just shifts horizontally or vertically by a fixed distance each time. If the image captured by the sliding window is determined to be a wall in stage 1, or to be a switch in stage 4, the sliding window will shift to its right by half of the sliding window width, i.e. 50 pixels in our experiments.

4. Results and Discussion

We tested our algorithm on various images obtained from the interior of buildings. Our proposed 3-stage detector eliminates most of the non-switch areas on the full image so that the image patches that are passed to the classifier (stage 4) have a higher probability of being switches (see Figure 3).

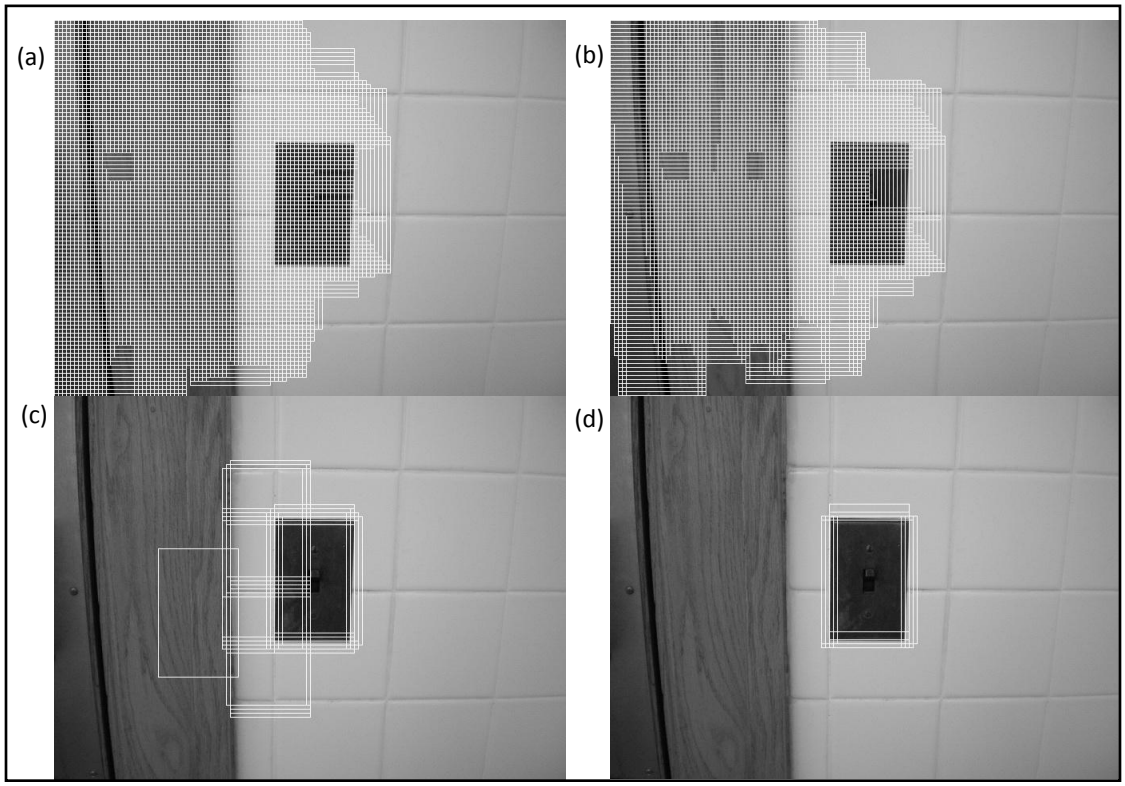

Figure 3 The white rectangles shown in this figure are bounding boxes for the positive image patches that are allowed through a stage of the detection algorithm. (a) "Positive" image patches allowed through the wall elimination stage (b) Image patches allowed through the symmetry detector stage (c) Image patches allowed through the frame detection stage (d) Image patches detected as switches by the final classifier

The number of image patches that survive each detection stage depends on the objects in the image, but for a wall like the one shown above, the image patches remaining after each stage are 2500, 1230, 42 and 11, or 30%, 15%, 0.5% and 0.1% of the original number of image patches respectively. The percentage improvement is the most obvious in the frame detection stage, establishing the importance of the rectangular frame characteristic for identifying a switch. The latter two characteristics could also be used as features to train the classifier, which would eliminate the problem of having to manually set the sensitivities and thresholds for each detection stage. An alternative solution would be to use k-fold cross validation errors to tune the sensitivities.

Full detection runs on other images managed to identify most of the switches, both individual switches, and switches that are part of a double or triple switch. On a test set of 15 full images containing at least one individual switch, our precision rate was 83%, and recall rate was 87.5%. The switches that were not detected were part of a multiple switch, and were not as "clean" because of extensive reflections, or had only two edges in the sliding window and additional noise from objects attached to the switch plate (See Figure 4). Further pre-filtering of the training and test images to remove more noise and reflections than a Gaussian filter can, and to adjust for lighting would probably improve our results.

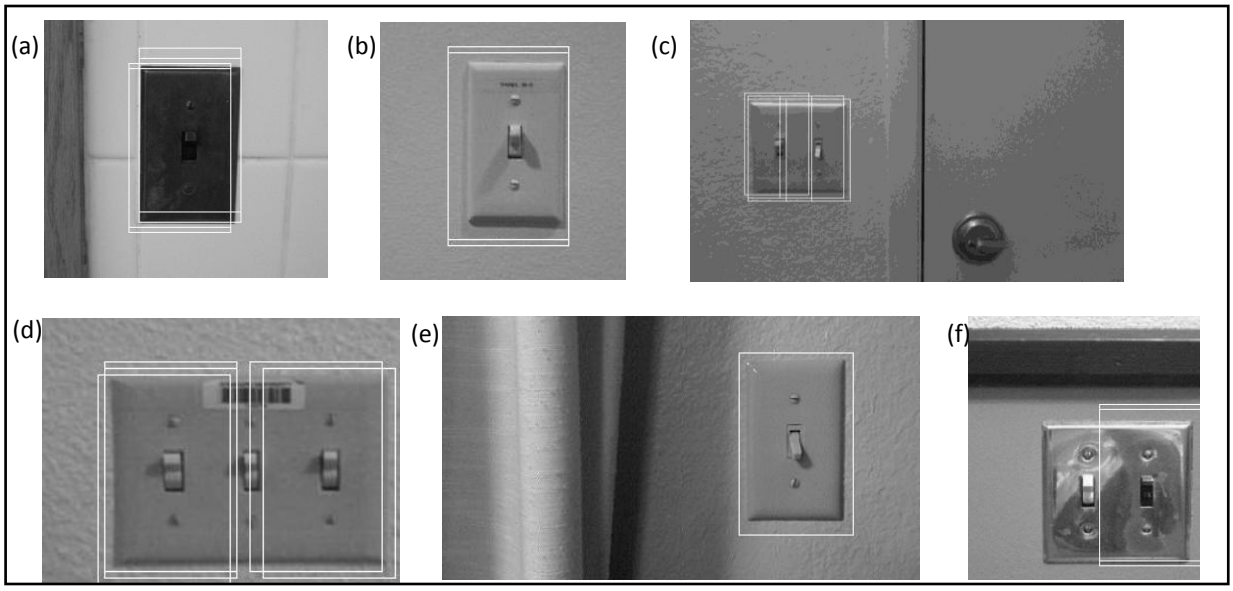

Figure 4 Some results of detection on full images. The white boxes show image patches that were detected as switches. (d) The switch in the middle was not detected because of the complicated barcode on top. (f) The switch on the left was not detected as it could not pass the asymmetry detection stage due to extensive reflections.

5. Conclusion and Future Work

Through our experiments, we conclude that feature extraction is crucial for increasing precision for light switch detection. Our detection algorithm detects most switches accurately, but by including other features such as Haar features, the symmetry of the image patch, and the probability that the image patch contains a frame, our results could improve further. Not all training examples have the same importance in training too, so we could consider using weighted training examples to improve the accuracy of the SVM classifier.

In addition, to take into account the possibility of non-linear separation between switches and non-switches, we could use a non-linear kernel for SVM classifier. While SVM is a powerful learning algorithm by itself, we could improve the performance of the detector by combining different learning algorithms using boosting.

6. Acknowledgements

We would like to thank Ellen Klingbeil for her patient guidance throughout the course of the project, and Professor Andrew Ng and the TAs for teaching us machine learning concepts.

7. References

[1] Y.Mae et al. Proposal of a Wheelchair User Support System Using Humanoid Robots to Create an SSR Society. (2003)

[2] P.Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. Conference on Computer Vision and Pattern Recognition (2001)

[3] R. Lienhart and J. Maydt. An Extended Set of Haar-Like Features for Rapid Object Detection. IEEE ICIP (2002)

[4] E. Klingbeil, A. Saxena, A. Ng. Learning to Open New Doors. RSS Workshop on Robot Manipulation (2008)

[5] E. Klingbeil, B. Carpenter, O. Russakovsky, A. Ng. Autonomous Operation of Novel Elevators for Robot Navigation. ICRA (2010)