ScrubBot: An Approach Towards Learning Stains

Anthony Chen and Detian Shi

***Abstract*— In this paper we investigate the application of robotics to the domestic chore of using a sponge to wipe off stains on a table after a meal. We divided the problem into two main components; stain detection and cleaning kinematics. For the stain detection sub-problem we mainly focused on developing algorithms that would enable a robot to correctly identify stains. We first developed and experimented with a series of methods to extract features from stains in images. To increase accuracy we then use machine learning techniques to classify stained sections on images. The developed solution detected stained areas in images with a reasonable degree of accuracy. The stain detection solution was then used to develop a process for the PR2 robot to scrub tables. Experiments run in a virtual environment show the PR2 scrubbing a stained area on the table.**

I. Introduction

While domestic applications of robots have been highlighted in popular culture practically applying robotics to accurately perform household tasks remains an open and interesting problem. In this paper we investigate the application of robots to a particular household task, specifically the task of using a sponge to wipe off stains on a table after a meal.

II. Related Work

Although very little has been done in terms of general stain detection, there has been some work in specialized instances of stain detection. Patel et al. [2] achieved a high degree of accuracy in blood spot, dirt stain and crack detection on poultry eggs using artificial neural networks. Similarly, Mertens et al. [4], examined the use of computer vision algorithms in detecting stains on eggs with some success.

In general, however, these works focus on a specific situation, with extensive prior knowledge about the scene and controlled conditions under which the algorithm is expected to work. In this project, we consider the problem more generally.

III. Approach

We divided the problem into two main components; stain identification and cleaning kinematics. For the stain identification sub-problem we mainly focused on developing algorithms that would enable a robot to correctly identify stains. First, we clearly established our definition of a "stain" as a discoloration produced by foreign matter having penetrated into or chemically reacted with a material. The most important property of these stains is the discoloration of the stained area from the background texture color.

Fig. 1. PR2

A. Image Segmentation

Since a stain will usually stand out in comparison with the background, one natural approach we explored is to segment the image into background and stain parts using standard segmentation techniques. To accomplish this, we used a watershed algorithm to segment the image. Specifically, watershed by flooding [1] was utilized. The segmentation process initially erodes the image to obtain seed points, which are defined as pixels whose values are representative of their neighbors. The watershed algorithm is then run on the seed points to obtain a segmentation. The initial seed points allow distinct segments to be treated contiguously and to partition the output in two, as a normal segmentation might produce many layers of segments. The watershed algorithm was chosen because of its speed and ability to clearly segment trivial test cases.

B. Average Windows

After initial testing, the initial watershed segmentation turned out to be too sensitive to background textures. An alternative method, referred here as the *average window* approach, was developed. In the average window approach, the image is converted to grayscale and discretized into fixed size square blocks or *windows* and the average window is calculated. For a window size of 10x10, this yields a vector of length 100 which we use as the baseline. Next, we calculate distance of every window in the image from this baseline window and normalize by scaling each distance with respect to the maximum and minimum window distance.

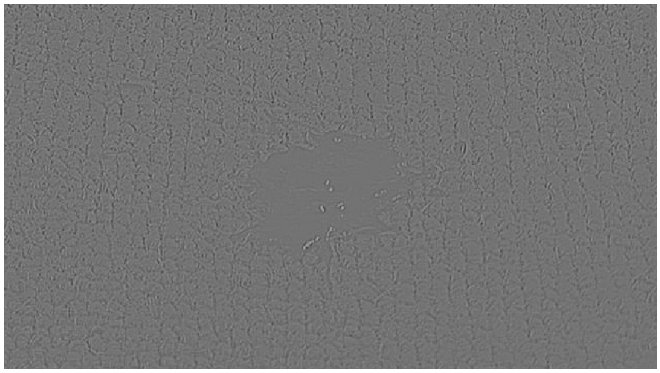

Fig. 2. Shitty Edges

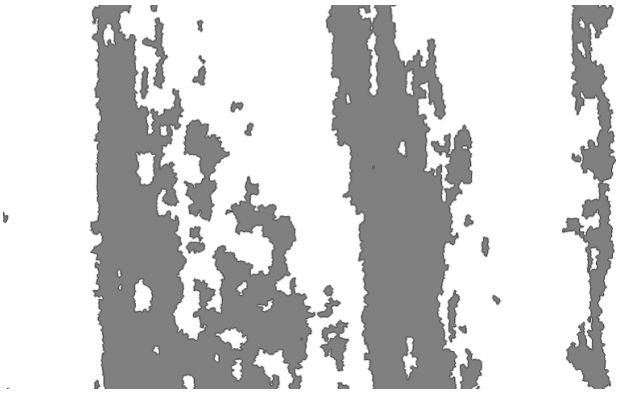

Fig. 3. Segmentation fail

Fig. 4. Segmentation fail

The distance metric used in this paper was L2-Norm. Finally, windows with a distance greater than a predefined threshold are then classified as stains. The motivation of this average window approach is that although each window may differ from some background texture, a window with a stain in it will differ more relatively to those that do not. In this way, we account for the differences created by a background, especially any repeating one.

C. Learning Features

A natural extension to the above approach is to apply classical Machine Learning methods to improve robustness. Rather than hand specifying how thresholds should be specified, we can learn them through supervised training methods. For this, we choose SVM's for its extensibility and effectiveness. For the features, we first split the image on a per channel basis, one for each of the three color channels. For each channel, we used the absolute difference between the average intensity in the window and the average, mean and mode intensity (per channel) of the image as features. In addition, the difference in variation of intensities within the window as well as the *edginess* of the window were also used as features. To obtain a value for *edginess* the image L is convolved with an edge detecting filter, as described in equation 1 and the resulting convolved image is similarly discretized into the same size windows. The edginess value for a window is determined by summing the pixel values of all pixels in the window. Figure 2 demonstrates the edge filtering.

$$L_{edges} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} * L \tag{1}$$

The motivation for the variance is that a stain might cover up an otherwise colorful background, so a decrease in variance might tell us something about whether the window is indeed a stain. Similarly, a window with a stain will have different degree of edginess as compared to those without.

IV. Perception Results and Discussion

The complete dataset used to evaluate the algorithms contains about 70 images, which results in roughly 70,000 training examples (at a window size of 10x10). To ensue that distribution of positive and negative examples is roughly evenly split, we took a simple random sample of the negative training examples. For every picture, we divide it into rectangular regions as governed by a window size parameter. These rectangular regions are used to classify windows as stained or non-stained. If a window overlaps with a labeled stain region, that window is considered to be correctly identified.

A. Segmentation

Several disadvantages were discovered through experimentation with the watershed segmentation. The watershed segmentation approach was unable to handle patterned backgrounds, blemishes and textures such as wood grain, as figures 3 and 4 demonstrate. Futhermore, upon segmentation it was often difficult to determine which portion of the mask was the stain and which was the background. Additionally the method was hard to modify, and therefore improve, because any changes could only be done through pre and post processing. These initial limitations suggested general limitations that any naive segmentation approach towards stain detection would suffer from. The established definition of a "stain" is general enough to include backgrounds of differing materials and as a result the effectiveness of segmentation suffers.

B. Window

Through experimentation it was qualitatively determined that in general the window approach performed more accurately than the initial segmentation approach. Figure 5 shows the result of running the window algorithm on a stain. While the window approach performed better than the segmentation approach, the results still varied on an image by image basis. Certain parameters of the algorithm could be adjusted to improve performance but the adjustments varied between pictures and cannot be set automatically. Table I displays the statistical results of the window detection approach. One thing to note is that the recall rate may not accurately reflect true values since they were calculated based on windows. A higher recall on larger windows is misleading since a stain might only encompass only part of the window.

Window Size	Average Precision	Average Recall
5	0.640206848	0.161015766
10	0.625077057	0.203232955
20	0.646491238	0.262179909
50	0.645036666	0.405706819

TABLE I
WINDOW RESULTS

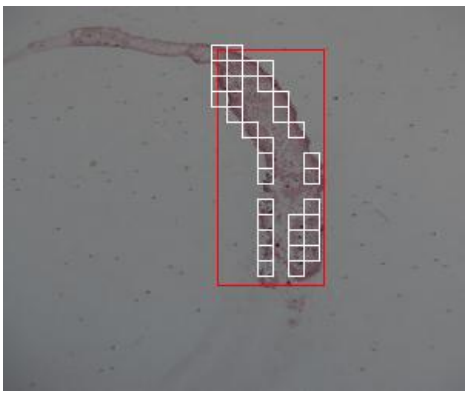

Fig. 5. Window-based

C. SVM Learned

The SVM implementation used in this paper was SVM-light [5] and the window size is 10x10 for the examples used in this paper. Table II displays the result statistics of training on a sub-dataset labeled as "dataset 1" and testing on a sub-dataset labeled as "dataset 2". Table III displays the result statistics of the reciprocal training and testing relationship. It is worth noting that the overall performance is higher in table III for multiple reasons. Dataset 1 is the first set of test images obtained and manually classified. The dataset contains images with slight contrast issues which leads to bad training data. This issue of contrast is further discussed later in this paper. Another reason for the poorer performance when training on dataset 1 results from misrepresentation of stains due to poor manual classification. Because most stains are not square in shape, attempting to classify curved

c	Accuracy	Precision	Recall
0.00075	46.2	35.81	9.58
0.001	50.71	51.88	19.55
0.01	55.52	55.19	58.76
0.1	45.67	46.77	62.69
1	50.94	63.67	4.37
10	49.02	48.71	36.99
100	49.02	48.71	36.99
1000	49.02	48.71	36.99

TABLE II
SVM RESULTS FROM TRAINING ON DATASET 1 AND TESTING ON 2

stains using square bounding boxes will inevitably result in areas misclassified as stains. As a result, many pictures in the "dataset 1" contain misclassified examples and this source of error was noted before "dataset 2" was classified. The bounding boxes in "dataset 2" are drawn much closer to the shape of the stain in order to reduce error and the results show a definite reduction.

Qualitative comparisons between the SVM detected and window detected regions also highlight the improved performance brought on by machine learning. Figure 6 shows the results of window-based detection on the left and the results of learning on the right. It is worth noting that not only does the learning approach label much more of the stain, it also correctly identifies sections of the stain that were not classified initially. Figure 7 best highlights the advantages of the learning approach. In the top image, the window-based detector incorrectly identified the background pattern as a series of stains. In the bottom image the SVM correctly labels the stained region while not misclassifying the background pattern.

A parameter that must be specified when using a SVM is c, which determines the penalty for a misclassified instance. Testing on the validation set, we found that $c = 0.0001$ generally gave good performance. This also correlates to a trade-off between recall and precision. By using a higher c, we trade higher recall for precision.

D. Error Analysis

Most of the cases where the SVM failed were in images where there are lighting differences or other artifacts that fit our original definition of a stain. Figure 8 shows a representative case of where our method fails. The image has major lighting differences which makes identifying the background difficult. This might be solved by first segmenting the image first and then applying the SVM on the individual segments. This will be explored for the following sprint.

E. Future Work

While the learning approach towards stain detection has proven to be more effective than segmentation and window comparison-based solutions, the process could benefit from additional image preprocessing to reduce error. For example, erosion filters would remove small noise and dilation filters would strengthen stain connected components. An additional source of error reduction is better manual classification

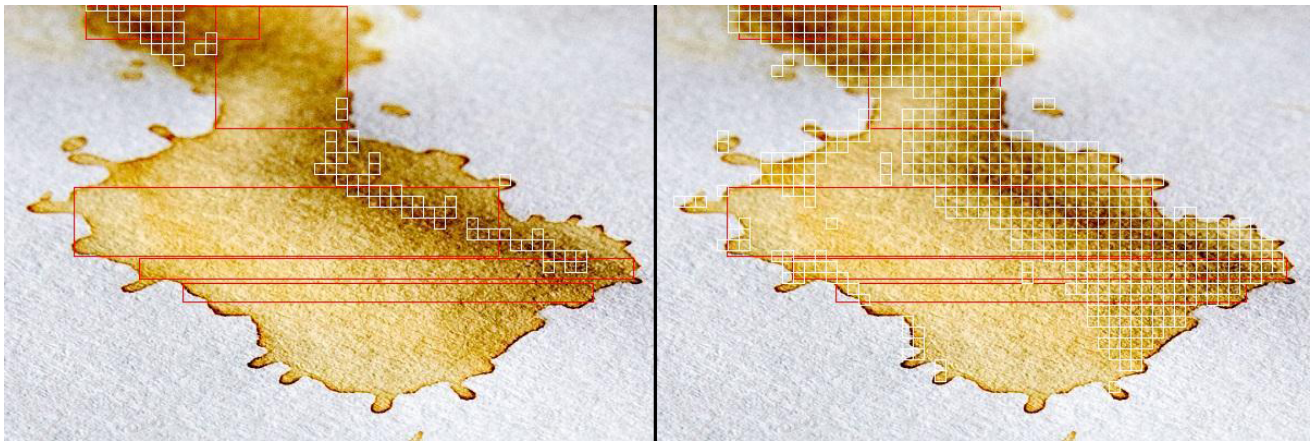

Fig. 6. Comparison of window feature detection (left) and SVM detection (right)

c	Accuracy	Precision	Recall
0.00075	66.12	72.74	51.62
0.001	66.12	72.71	51.65
0.01	66.12	72.77	51.59
0.1	49.99	50.01	87.9
1	50.19	50.15	70.8
10	49.61	49.82	97.86
100	56.28	54.26	80.15
1000	56.28	54.26	80.15
10000	56.28	54.26	80.15

TABLE III

SVM RESULTS FROM TRAINING ON DATASET 2 AND TESTING ON 1

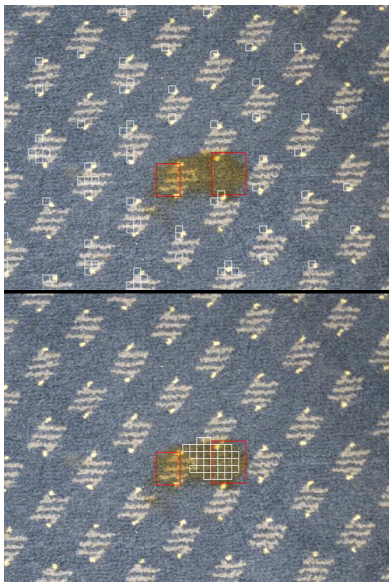

Fig. 7. Comparison of window feature detection (left) and SVM detection (right)

Fig. 8. Bad Contrast

techniques. As noted in the results section there was a significant improvement in performance when manual classification techniques were improved. While the creation of a classification tool may be out of the scope of the project, we may work on it if there is additional time.

V. EXPERIMENTAL RESULTS

We demonstrated our detection algorithm on the PR2 robot in a virtual environment using the Gazebo robot simulator. The robot was required to detect a stain on a table in the virtual environment and move its arm in a scrubbing motion over the stained areas. A stained tabletop image was applied as a texture on the top of the virtual table. The image applied was taken from the RGB camera on a Kinect sensor. The Kinect sensor device was head mounted on a PR2 robot and the robot position with respect to the table was similar to the arrangement in the virtual environment.

A. Virtual Hardware

Our virtual robot consisted of a PR2 with a head mounted Kinect sensor. The Kinect sensor can provide both 2D RGB images as well as 3D point cloud data.

B. Software

Using the OpenNI framework with the Kinect sensor we can register the point cloud with the RGB images and transform points between the 2D RGB image plane and the robot coordinate frames. The point cloud data is preprocessed using RANSAC to determine the dominant plane in the scene, which is the table. The resulting segmented point cloud is then used to create a mask for the 2D RGB image, which excludes objects on the table as well as areas of the image not on the table. 2D RGB images are then captured from the Kinect RGB camera and processed using the SVM window algorithm for stain detection discussed previously. Points on the 2D RGB image plane determined to be stained are transformed to the robot coordinate frame and kinematics processes move the robot arm and perform a scrubbing routine.

VI. Conclusions

In this paper, we discused a learning approach to general detection of stains. We focused on the crating of features that would describe stains, which by nature is an elusive category. We evaluated our results on an original dataset tagged by hand, showing great results for most but the hardest images. Finally, we ran our algorithm in simulation. Due to the limitations of the the simulator, we were not able to clean stains, although the movements appear to be correct.

Acknowledgments

We would like to thank Professor Saxena and the CS4758 teaching assistants for providing us with the opportunity, resources and knowledge to accomplish this project.

References

[1] Serge Beucher and Christian Lantujoul. Use of Watersheds in Contour Detection. In International workshop on image processing, real-time edge and motion detection (1979).

[2] Patel, V. C., McClendon, R. W., and Goodrum, J. W., 1998, Color Computer Vision and Artificial Neural Networks for the Detection of Defects in Poultry Eggs, Artificial Intelligence Review, v. 12, p. 163 - 176.

[3] Abid, K. (2012, July 11). stackoverflow Retrieved from http://stackoverflow.com/questions/11294859/how-to-define-the-markers-for-watershed-in-opencv/11438165

[4] Mertens, K. (2005). Dirt detection on brown eggs by means of color computer vision. Manuscript submitted for publication, Egg Quality and Incubation Research Group, , Available from US National Library of Medicine. (16335136)Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/16335136

[5] 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.