ScamCanner

Agni Purani, Neha Dalmia, Rishabh Agarwal, Rohit Raj, Rohit Sutradhar, Satwik Chappidi*

***Abstract*— ScamCanner is an Adobe Scan/Cam Scanner like software which is used to produce scanned documents from camera images, *i.e.*, given an image of a document, in varied orientations and lighting, it identifies the region of interest of the document in terms of edges and corners and correctly orients and displays the image as a scanned copy of the document. It was a project that meant to study different feature detectors and use logistic regression to create a trained model that helps in converting images of documents into a proper scanned document. For this, we use SURF to detect features and apply a Hough Transformation based implementation to obtain a bounding box. This is followed by homography and thresholding to make the image clearer and give it the scanned document feel.**

I. INTRODUCTION

The use of mobile phones is increasing, already being at 2.9 billion worldwide in 2019 and expected to rise to 3.5 billion by 2023 [1]. In many highly developed countries, the number of smart phone users is very high in proportion such as countries like South Koera(82.78% by 2022) [2] and United States(81% in 2019) [3]. Thus the need of portable solutions for things like scanner is an absolute necessity. It may seem that in such a busy world, people may not have enough time to even wait to take a scan. Taking images of the document may seem as a good idea but this makes it hard to read anything on it and unnecessary background information and noise is introduced. This gives a lot of challenges as the lighting condition can vary, the orientation of the document may not be proper in the image and a small part may be outside of the image. ScamCanner offers such a solution by using simple logistic regression to convert any image of a document into a easily readable image that looks like it has been scanned.

II. PROBLEM STATEMENT

The task involves developing a scanning software which produces scanned copies of documents given an image in varying orientation and lighting conditions. The task is further divided into 6 sub-tasks:-

A. *Data-set Collection and Annotation*

Data-set collection involves collecting a number of training examples which widely describe the various conditions the software might come across in daily usage. Each training example is ideally an image of an A4 sized document with simple text, to be captured in different lighting and orientation.

Annotation involves running the feature detector on each training image and manually classifying which feature represents what.

*Names written in alphabetical order

B. *Feature Detection and Feature Descriptors*

For feature detection, we have to run a feature detection algorithm of our choice and pick the top N features, N being a parameter we decide. After picking the top N features, we extract their corresponding descriptors.

C. *Logistic Regression*

After collection and annotation, we have to train a linear classifier to identify whether a feature is a relevant/boundary feature, *i.e.*, whether it lies inside the final document, or irrelevant, *i.e.*, outside the defined boundaries. The boundary features can be defined using edges or corners, whichever produces better results.

D. *Approximating a Region of Interest*

After detecting the features, we have to identify a Region of Interest. We approximate a bounding shape around the features and use that to construct an appropriately sized rectangle or bounding lines which covers the maximum area of the region of interest of the document.

E. *Homography*

Once an appropriate bounding rectangle is obtained, we have to apply Homography in order to correctly orient the resulting image.

F. *Color Spaces and Thresholding*

After the document is oriented correctly, we have to manipulate the output by adjusting the colour spaces and applying thresholding to make the document as clear as possible.

III. RELATED WORK

A. *Document Scanning Using Multiple-Pass Mosaicking*

The system compares each pair of consecutive images and derives motion parameters that indicate the relative motion between each pair of consecutive images. The system utilizes the derived motion parameters to align and merge each image with respect to the previous images, thereby building a single, mosaic image of the document. In the illustrative embodiment, the motion parameters are derived by minimizing a sum of squared differences equation on a pixel-by-pixel basis. [4]

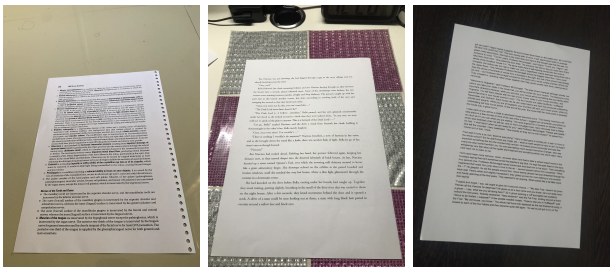


Fig. 1: Some examples of images taken

B. *Robust and Fast Skew Detection Algorithm*

A robust and fast skew detection algorithm based on hierarchical Hough transform is proposed. It is capable of detecting the skew angle for various document images, including technical articles, postal labels, handwritten text, forms, drawings and bar codes. The algorithm is robust even when black margins introduced by photocopying are present in the image and when the document is scanned at a low resolution of 50 dpi. [5]

IV. INITIAL ATTEMPTS

ScamCanner had gone through several major and minor changes before its final version. Also the way the first version of the program worked was very different from what it is currently. The following subsections describe what was experimented and the better performing algorithm or part of the program was used to make the final version.

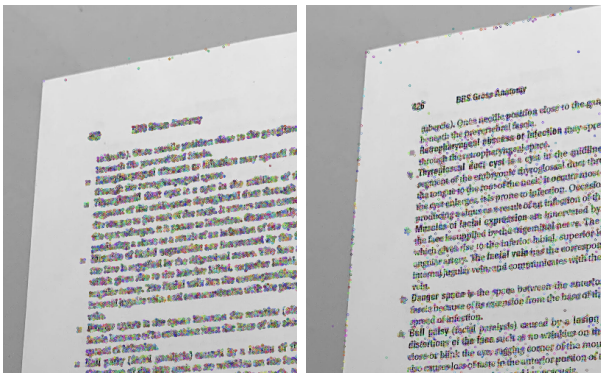
A. *Data-set Collection*

629 pictures of documents pictures were taken. Each page of the document were taken with different backgrounds (plain, floral, glass, on objects like keyboard and certain other patterned background), different lighting conditions (with flash in dark room, bright light, single point source light) and with different orientations as shown in Figure 1.

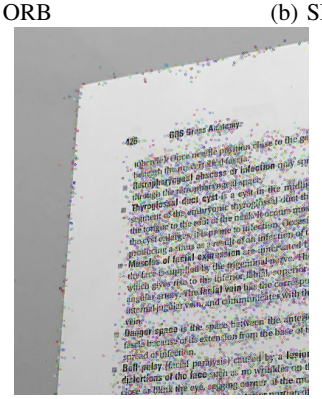
B. *Feature Detection*

The feature detection is an essential part of the project as the feature that are to be detected are to be used for the purposes of training and so three feature detectors were short listed - ORB [6], SIFT [7], SURF [8]. Even though ORB is efficient compared to SIFT or SURF [6] the features ORB detects are more towards the center of the images compared to the SIFT and SURF [9] which is not appropriate for this project as we plan to find the document by figuring out the edges and the corners in the image. Between SURF and SIFT detectors SIFT is a faster one but it doesn't deal with noisy images as good as SURF and so SURF seems to go better with the images that the program would be dealing with [9]. To confirm with the what was already found a small comparison was done as shown in Figure 2.

The SURF feature detector was used in the initial attempts without applying any processing to the images except resizing all of them to a particular height or width depending on the orientation (portrait or landscape) as some images were



(a) ORB (b) SIFT



(c) SURF

Fig. 2: Comparison of different feature detectors of the same image at a corner. In this case SURF detector finds a better ratio of boundary features compared to the others.

of low resolution ans some were really high to create uniformity. Not applying resizing would cause smaller images to given less number of features, giving a small data set to train, and large images would give overwhelming amount of data that would take a lot of time annotating.

C. *Data Annotation of the Descriptors*

For data annotation the first approach was to manually label points one by one with a prompt asking to label a feature shown in the image '1' (a boundary point) and '0' (not a boundary point). Two data sets were prepared for training, one where '1' meant that it is a corner and the other data set it meant an edge. We tried classifying the top 100 features based on the score and labelled them positive or negative. Later one a better way to label points was found by creating an interactive program in which the points around the corner or along the edge could be selected by dragging the cursor over them. This significantly speed up the annotation time.

During annotation, the number of positive points that appeared in top 100 features where not much ($\approx 8\%$ for corners and negligible amount for the edges) of all the features that appeared. To solve the problem, all the features were taken and all the positives were selected but only the top 100 negatives were taken and the rest were ignored. Another solution that existed was to apply filters. As two sets of data

Method	Corners	Edges
Logistic Regression	96.8%	63.1%
Logistic Regression using SMOTE	94.5%	62.9%
Support Vector Machines	97.6%	63.9%

TABLE I: Accuracies of the test data on different methods and data sets.

was supposed to be made for training, different filters should be used as we look out for different filters. For the corners applying filters did not make much of a difference and so the first solution was used for the corners. For the edges a few filters where applied. The median filter is good at removing thin lines and small noise without blurring the shape of big objects of the image [10]. So, median blur was used to get rid of the text. To increase the number of edge features a sharping filter was applied

At the end two csv files (one where corners were labelled and. the other were edges) were labelled were generated with 66 columns each - one for label, second for keypoint ID and the rest 64 were feature descriptors.

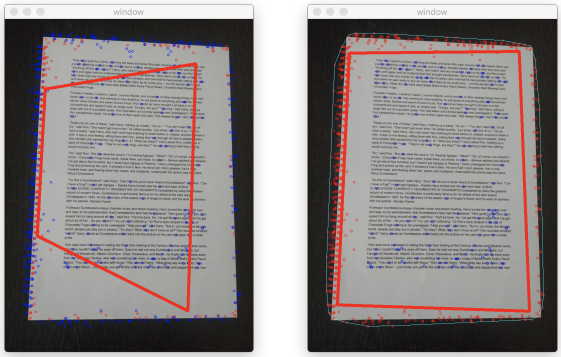
D. *Training*

The training is the essential part of the project. The raw data was used to train the data we had generated. We tried out there different methods for both the data sets and the accuracy for each is shown in Table I.

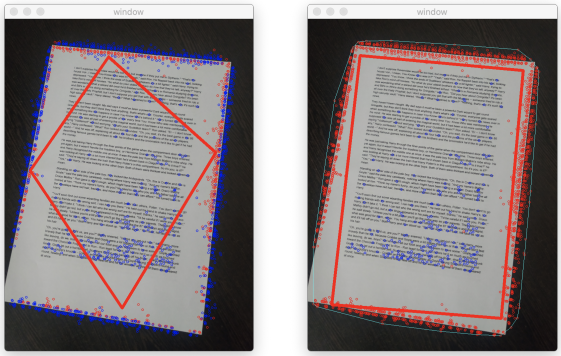
E. *Finding the Region of Interest*

The next objective is to find a bounding quadrilateral for the paper that lies in the image. In this case it might seem that the corner data might perform the best. Even though the accuracies of the training for corners may be high, it is not the only factor it depends on. After running an example we found that it was not great as we found some corners were missing and some images had multiple features for a single corner. To resolve the later mentioned issue, K-Means clustering algorithm can be used to find four different clusters which can represent a corner. But, the first issue cannot be fixed. Moreover, it was found that the edges trained data with around 63-64% accuracy outperformed the corner trained data as shown in Figure 3.

This surprising result was because in case of corners, there are a very few features that can be considered as corner (maybe 8 out of 10,000 features). So, whenever we test the accuracy it may be high but even missing a corner or adding a few unnecessary corners features may cause it to fail. In case of the edges, there can be a lot of features and even if some of them are missing, it can still be figured out that it is an edge. Also, if a few points are falsely detected positive they can not give much weight-age compared to tens or hundreds of points on the edges. To detect the bounding quadrilateral of the document using the edge classified features, a convex hull was found for all the edge features that was classified by the trained model. This resulted in a convex polygon with four or more sides. To reduce it to a quadrilateral, a possible



(a) Left: Using Corners. Right: Using Edges.



(b) Left: Using Corners. Right: Using Edges.

Fig. 3: Figure 3a shows a standard comparison between both. Figure 3b shows a comparison where one corner is slightly outside. It is clearly shown that in both cases edges finds a better bounding quadrilateral

solution was to find a rotated rectangle with minimum area that encloses all the points of the convex hull. Also, to make it a better fit, it is a better idea to reduce the size by pulling it inwards towards the features rather than enclosing all the features but taking Gaussian weight for each point.

F. *Transforming Perspective*

The final step was to apply perspective transform so that the paper or document looks rectangular. Later on, thresholding can be applied to convert it to a binary image to give a scan like feel. Intially, a global thesholding was used but in pictures where shadows appeared it failed and so adaptive thresholding was used.

V. FINAL APPROACH

Data-set Collection and Annotation: We used 600 pictures to formulate our data-set. We first convert the image to gray-scale, followed by dilation then erosion. This was followed by sharpening and contrast limited adaptive histogram equalization. Then we applied **SURF** algorithm to extract the key-points of the modified image. Each key-point comprises of 64 descriptors and a vector storing the location of the key-point. For annotation we printed all the key-points and using mouse clicks we identified regions near the edge. All the

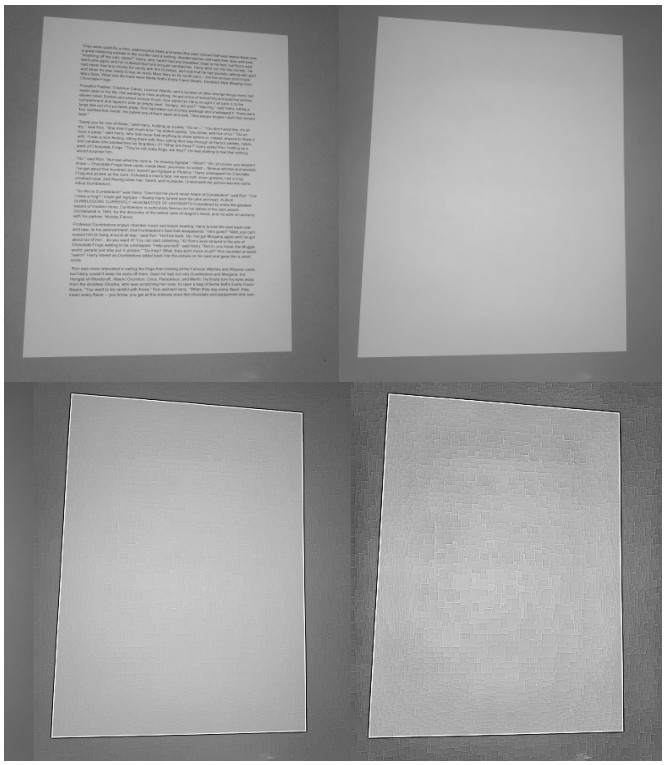


Fig. 4: Stages of filtering which include applying meanshift filter for blurring the background and textual content of image and then sharpening the image to make edge features more prominent

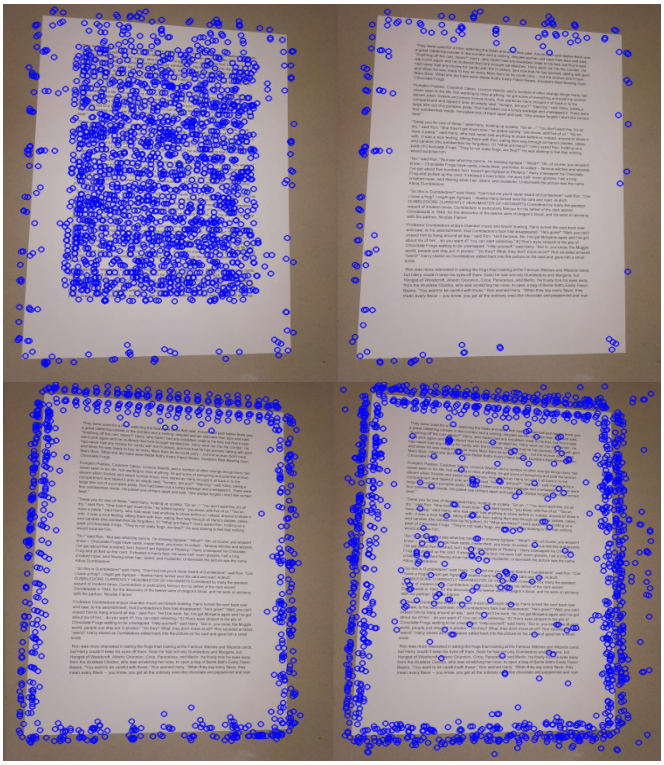


Fig. 5: Detected feature points at different stages of filtering the image

key-points lying in this region are marked as positives. We take all the positive features and only the negatives which are included in the top 2000 features. All the key-points are added to a csv file, along with their location vector, descriptors, and a column in which they are marked with a '1' if they are edge points and '0' if they are not.

Logistic Regression: To train the logistic regression model, we took all the positive features and only the negatives which appear in the top 2000 features. We then used SkLearn to perform logistic regression dividing the training and testing data in the ratio 3:1. We obtained a positive-negative ratio of approximately 1:3 with an accuracy of 90.24%.

To extract the document from the image, we first apply the same filter that was applied while the annotation of the dataset, and then use the SURF feature detector to get the key-points. Then the trained logistic regression model is applied to classify the features into edges and not edges. To filter out any stray false positives we check if there are features present in the neighbourhood of the given feature. The remaining edges features are then used to approximate the edges. For this we used a **Hough Transform** based implementation.

Hough Transform: Each feature votes for values of line parameters r and θ in the Hough space which is a 2-d array. Then the local maximas of the 2-d array are found. These maximas are sorted on the basis of the number of votes they receive. Close lying peaks are filtered out, as they represent

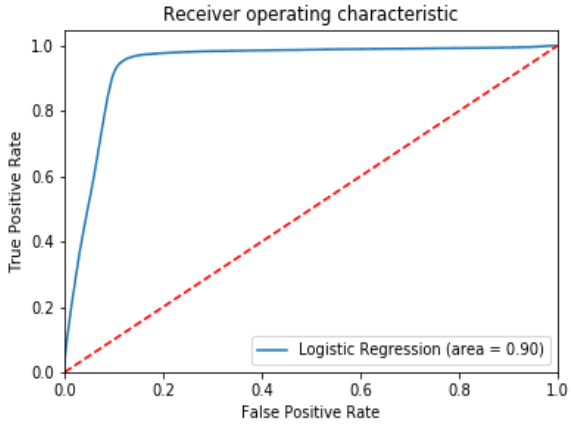


Fig. 6: Graph showing the count of false positives

lines along the same edge. This is done by checking if they lie within a circle having a certain radius. We separately need to filter out lines representing the same edge that pass through the origin as they won't necessarily be represented by close lying peaks. This is done by checking if r is around 0 and the difference in the values of θ lie within a certain limit.

Finding Corners: The edge lines returned by the HoughLines function may not represent the 4 edges. There might be some redundant lines present along as well. So we take the 4 lines with the most number of votes as the edges and take their points of intersection as the corners. The corners

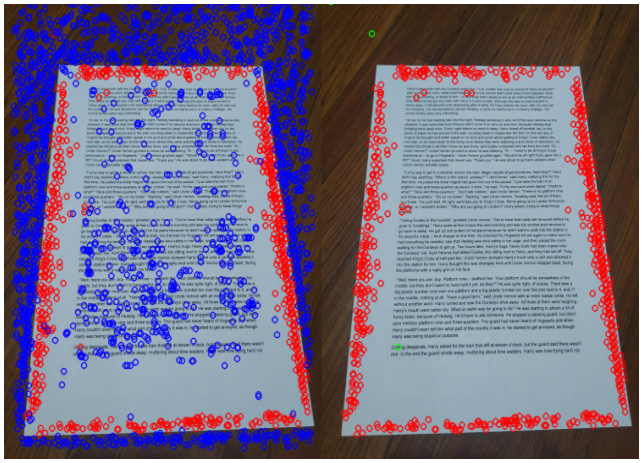


Fig. 7: Points classified as True positives(in red), False positives(in green), negatives(in blue)

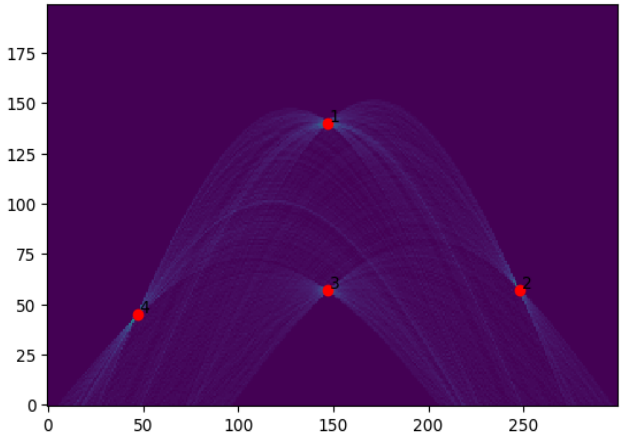
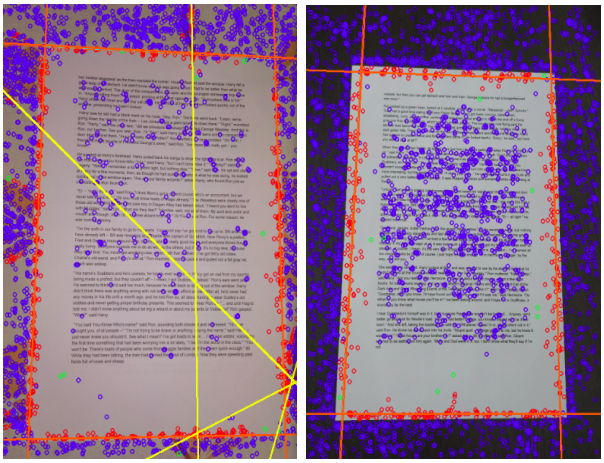


Fig. 8: Sinusoids corresponding to all possible lines, formed by positive points, in the image and the final parameters for four lines obtained(red points) for a given image

are then passed to the transform function for final document extraction.

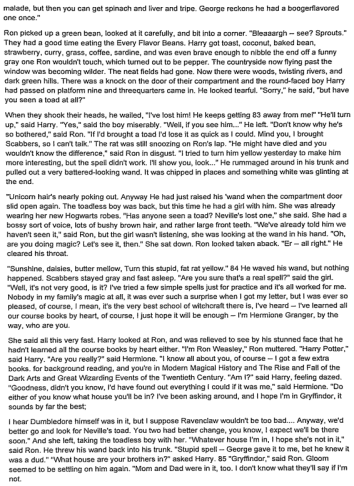
Perspective Transform and Thresholding: We order the four points we obtained, and obtain the largest width and height and create a separate list of four points. We use cv2.getPerspectiveTransform to obtain the Transformation matrix M and then cv2.warpPerspective to get our final birds-eye view image using this matrix following by resizing of the image. We then apply **adaptive thresholding** to convert the image to a black and white form. Adaptive thresholding is used as the image can contain shadows, stray text, etc. After doing all this, we finally manage to obtain a resized, thresholded bird's eye view using the four points we had extracted.

Errors encountered and their solution: A large number of features were getting detected in the text area. So we apply dilation followed by erosion to remove the text. If sufficient no of features are not detected along the edges the HoughLine function returns erroneous lines. So in order



(a) (b)

Fig. 9: Figure 9a shows all lines detected by hough transform where red lines represent edges of quadrilateral and others are shown in yellow. Figure 9b shows final quadrilateral formed by lines detected in the process.



malade, but then you can get spinach and liver and tripe. George reckons he had a boogerflavored one once."

Ron picked up a green bean, looked at it carefully, and bit into a corner. "Bleaaargh — see? Sprouts." They had a good time eating the Every Flavor Beans. Harry got toast, coconut, baked bean, strawberry, curry, grass, coffee, sardine, and was even brave enough to nibble the end off a funny gray one Ron wouldn't touch, which turned out to be pepper. The countryside now flying past the window was becoming wilder. The neat fields had gone. Now there were woods, twisting rivers, and dark green hills. There was a knock on the door of their compartment and the round-faced boy Harry had passed on platform nine and threequarters came in. He looked tearful. "Sorry," he said, "but have you seen a toad at all?"

When they shook their heads, he wailed, "I've lost him! He keeps getting 83 away from me!" "He'll turn up," said Harry. "Yes," said the boy miserably. "Well, if you see him..." He left. "Don't know why he's so bothered," said Ron. "If I'd brought a toad I'd lose it as quick as I could. Mind you, I brought Scabbers, as I can't talk." The rat was still snoozing on Ron's lap. "He might have died and you wouldn't know the difference," said Ron in disgust. "I tried to turn him yellow yesterday to make him more interesting, but the spell didn't work. I'll show you, look..." He rummaged around in his trunk and pulled out a very battered-looking wand. It was chipped in places and something white was glinting at the end.

"Unicorn hair's nearly poking out. Anyway He had just raised his 'wand when the compartment door slid open again. The toadless boy was back, but this time he had a girl with him. She was already wearing her new Hogwarts robes. "Has anyone seen a toad? Neville's lost one," she said. She had a bossy sort of voice, lots of bushy brown hair, and rather large front teeth. "We've already told him we haven't seen it," said Ron, but the girl wasn't listening, she was looking at the wand in his hand. "Oh, are you doing magic? Let's see it, then." She sat down. Ron looked taken aback. "Er — all right." He cleared his throat.

"Sunshine, daisies, butter mellow, Turn this stupid, fat rat yellow." 84 He waved his wand, but nothing happened. Scabbers stayed gray and fast asleep. "Are you sure that's a real spell?" said the girl. "Well, it's not very good, is it? I've tried a few simple spells just for practice and it's all worked for me. Nobody in my family's magic at all, it was ever such a surprise when I got my letter, but I was ever so pleased, of course, I mean, it's the very best school of witchcraft there is, I've heard — I've learned all our course books by heart, of course, I just hope it will be enough — I'm Hermione Granger, by the way, who are you.

She said all this very fast. Harry looked at Ron, and was relieved to see by his stunned face that he hadn't learned all the course books by heart either. "I'm Ron Weasley," Ron muttered. "Harry Potter," said Harry. "Are you really?" said Hermione. "I know all about you, of course — I got a few extra books. Background reading, and you're in Modern Magical History and The Rise and Fall of the Dark Arts and Great Wizarding Events of the Twentieth Century. "Am I?" said Harry, feeling dazed. "Goodness, didn't you know, I'd have found out everything I could if it was me," said Hermione. "Do either of you know what house you'll be in? I've been asking around, and I hope I'm in Gryffindor, it sounds by far the best;

I hear Dumbledore himself was in it, but I suppose Ravenclaw wouldn't be too bad.... Anyway, we'd better go and look for Neville's toad. You two had better change, you know, I expect we'll be there soon." And she left, taking the toadless boy with her. "Whatever house I'm in, I hope she's not in it," said Ron. He threw his wand back into his trunk. "Stupid spell – George gave it to me, bet he knew it was a dud." "What house are your brothers in?" asked Harry. 85 "Gryffindor," said Ron. Gloom seemed to be settling on him again. "Mom and Dad were in it, too. I don't know what they'll say if I'm not.

Fig. 10: Final output after applying homography and changing to suitable colourspace

to increase the features along the edges a sharpening filter is applied. For dark images and images with background color similar to that of the document the edge features were not detected properly. So histogram equalization is used to increase the contrast. In certain images, a complicated background resulted in false positives. To remove them we check for other features in their neighbourhood and if they are below a certain threshold we discard them. While making the Hough space no restrictions were placed on the values of r as a result of which negative values of r got mapped to wrong values. So limits were imposed on the values of r.

VI. RESULTS AND OBSERVATIONS

Dataset collection: Feature detection and annotation was used to make dataset and it was done for all done for all images twice, once considering only corners as positives and secondly, considering all points lying along edges positives. Following data was obtained: -

Label	Method 1	Method 2	Final Method
1	4587	28187	117744
0	57172	29641	367448

TABLE II: Positive and negative count

Thus, it can be observed that counting points along edges as positives gave more balanced dataset compared to other method.

Point classification using linear classifier: Logistic regression was used to make binary classification model for feature points detected in image. From the model, following confusion matrix was obtained: -

	0	1
0	83129	8863
1	2977	26329

TABLE III: Confusion matrix

For given test set, we got accuracy of 63.1% for edges which improved to 90.2% after applying filters and similarly applying logistic regression on corners dataset we got 96.8% accuracy. Even though the accuracy is higher in the case for corners we proceed with edges as it has a more balanced data set.

Determination of quadrilateral enclosing the document: Hough lines was employed to find the rectangle enclosing the document and the following results were obtained on few images: -

Thus, we observe that this made gave very good approximation of actual quadrilateral as very less area remains uncovered by the predicted quadrilateral and also extra area covered by quadrilateral was very less. The experiment was repeated for a big subset of image data set and following results were obtained: -

But there was some part of images in the dataset for which algorithm raised exception that either one edge is missing or vertex is missing and the common element between all those images was that either they actually has one or more corners missing or the background of image was too vivid and some false positives were detected in the background which affected the quadrilateral.

VII. FUTURE WORK

Feature points detection: For detection of feature points and balancing the count of detected positives, we used meanshift filter for blurring the text portion and background, followed by sharpening, but this was not sufficient as some false positives were still detected and this problem can be solved by using blurring in combination with morphological operations like **morphological gradient** while annotation and classification of points.

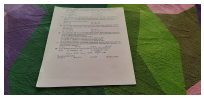
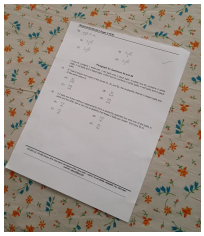
Image	Area recovered(%)	Extra area in detection(%)
	94.84%	0%
	89.37%	0%
	96.15%	0%

TABLE IV: Quadrilateral details for some images

Area recovered(%)	Extra area in detection(%)
93.31%	0%

TABLE V: Quadrilateral details for a large set of images

Boundary detection: For detection of suitable quadrilateral enclosing the document use of ranking SVM[12] and listnet[11] for assigning scores to each candidate quadrilateral and then use that score for finding best quadrilateral satisfying the purpose can be found. This method will be less vulnerable to false positives as our method for getting ideal rectangle is vulnerable to false positives.

Content of page: Currently, this method of boundary detection fails if the page contains images and tables and algorithm needs to be improved by using methods like **dhsegment**[13] to separate features from inside the page from features detected along boundary of page.

VIII. CONCLUSION

In this project, we compared different tools available for detection of feature points in a image containing text document and tried to use them to create robust boundary detector using binary classifier to classify feature points as boundary points and non-boundary points.

Finally, we concluded that SURF feature detector provided best results as other feature detectors failed to get sufficient positives despite blurring of background. Binary classification of feature points using logistic regression and usage of

Hough lines to make boundary out detected points gave best results out of all other options considered.

REFERENCES

[1] Statista. Number of smartphone users worldwide from 2016 to 2021. Available online: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (accessed on 10 May 2020).

[2] Statista. Smartphone penetration rate as share of the population in South Korea from 2015 to 2022. Available online: https://www.statista.com/statistics/321408/smartphone-user-penetration-in-south-korea/ (accessed on 10 May 2020).

[3] Pew Research Center. Mobile Fact Sheet. Available online: https://www.pewresearch.org/internet/fact-sheet/mobile/ (accessed on 10 May 2020).

[4] Frederic Dufaux & Sing Bing Kang & Robert Alan Ulichney. US7046401B2, United States Patent and Trademark Office, 16-05-2006. US7046401B2 - Camera-based document scanning system using multiple-pass mosaicking, https://patents.google.com/patent/US7046401B2/en

[5] Bin Yu & Anil K. Jain. A robust and fast skew detection algorithm for generic documents, Pattern Recognition (1996).

[6] Rublee, Ethan & Rabaud, Vincent & Konolige, Kurt & Bradski, Gary. (2011). ORB: an efficient alternative to SIFT or SURF. Proceedings of the IEEE International Conference on Computer Vision. 2564-2571. 10.1109/ICCV.2011.6126544.

[7] Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004).

[8] Bay H., Tuytelaars T., Van Gool L. (2006) SURF: Speeded Up Robust Features. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision – ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol 3951. Springer, Berlin, Heidelberg

[9] Karami, Ebrahim & Prasad, Siva & Shehata, Mohamed. (2015). Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images.

[10] Abdalla Mohamed Hambal, Dr. Zhijun Pei, Faustini Libent Ishabailu, "Image Noise Reduction and Filtering Techniques", International Journal of Science and Research (IJSR), https://www.ijsr.net/search_index_results_paperid.php?id=25031706, Volume 6 Issue 3, March 2017, 2033 - 2038

[11] Zhe Cao, "Learning to rank: from pairwise approach to listwise approach." ICML, 129-136, 2007.

[12] Thorsten Joachims. "Optimizing search engines using clickthrough data". SIGKDD, 133-142, 2002.

[13] Sofia Ares Oliveira, Benoit Seguin, Frederic Kaplan "dhSegment: A generic deep-learning approach for document segmentation" 2019