Sprint Evaluation

What is the progress of your project in this sprint? What goals are achieved? What problems are overcome? If you are updating your plans what are your justifications?

Character recognition and line detection are researched and implemented by Ögem Çetin. For training the part of the coupon containing the numbers played and recognizing the digits and characters in there, Tesseract OCR is used. Tesseract is an open source optical character recognition engine that converts a wide variety of image formats to text. In coupon samples PNG format is preferred. Although the results are mostly accurate, in some cases, a couple of erroneously recognized characters and digits are encountered, such as 'B' recognized as '8' and '11' as 'H'. To prevent this, the erroneously recognized characters and digits are corrected manually on Tesseract interface. When training and recognizing characters in multi-lined coupons, it is observed that Tesseract does not combine the lines and does not read as one line. On the contrary, the numbers played are recognized as in order of appearance. In detection of the lines around the part of the coupon containing the numbers played, Hough Line Transform is used. Since the lines are straight and horizontal The Probabilistic Hough Line Transform is applied. Character recognition and line detection are tested on 20 samples, 5 samples for each type of coupon.

Identification of the coupon type by logo related tasks are assigned to Ulas Dalli. Our first intention to identify the coupon logo was Histogram Image Comparison technique by calculating the histograms of the images. However, this approach was wrong since histogram technique calculates the number of pixel values for every intensity value in range [0-255] and our coupons basically consist of two colors, namely; black and white. Hence, in terms of the comparative intensity values of the pixels, it does not mean anything.
What to achieve to determine the coupon type is simply comparing two-color logo with the models that we have defined before and finding the one that has more similarities. Another reliable way to do is finding keypoints and comparing them. We follow three steps to identify the coupon logo:
1- Finding the keypoints on the coupon to be identified
2- Extracting the descriptors of keypoints
3- Comparing the descriptors of the logo to be identified with all logos' descriptors
In order to find similarities, OpenCV's FeatureDetector algorithms are researched and tried. First, SimpleBlobDetector algorithm which seems to be simpler than others is utilized to extract keypoints. This algorithm finds small areas in accordance with the supplied parameters on pictures. The basic problem with this algorithm is that it finds very few keypoints. SimpleBlobDetector decides whether an area will be a keypoint according to these parametes: circularity,color,pixel,convexity/concavity. After that ORB(Oriented BRIEF keypoint detector) is utilized in order to find keypoints. This algorithm finds its keypoints in a more complex way. This algorithm always returned with the correct result, however; it has far beyond features than what we need to compare logos, so we decided to use other Binary Descriptors. First intented BRIEF algorithm has no implementation in default OpenCV packets so we decided to BRISK which is faster and also reliable as well. Nowadays, optimization on the third step is being researched. The intended way to do is to save keypoint descriptor information of the logos to be compared in a file (such as txt,xml,yaml) and using this file for comparison. Extensive Testing on Coupon Samples (T7) task cannot be initiated yet, since the tasks that the testing task depends on have not been fully completed yet.

Perspective correction is implemented by Mesut Yilmaz. For perspective correction functionality, at the beginning there were two main methods that are considered to detect coupon borders,these are detecting coupon borders by line detection, and detecting coupon borders by finding contours on the image. Line detection method was decided to be applied with edge detection method. The goal was finding lines and checking whether there is an edge on that pixel which has line on it. This approach had some problems. Since we are taking the coupon photo from user, the coupon will always be wrinkled, much or less, because of that the coupon borders on the image, are not straigth, they are much or less curved. For this reason detected lines do not really represent the borders, but detected edges perfectly fits to coupon borders because, edge detection algorithm(Canny Edge Detection), does not care whether detected edges are straight or not. As a result of this, even detected lines and edges are close to each other, mostly they don't lay on same pixels. After examining example projects, finding contours on the image and getting quadral shapes were the most used method for detecting coupon borders to apply perspective transform. Because even the coupon borders are a little curved, it finds closed shapes and it detects four corners of that shape. To eliminate the performance issues of this method the input image is always resized to 600*800 matrix. This is also good for

possible memory issues for mobile phones. To improve the speed of the code, and performance Gaussing Bluring is applied to input image, this addition will blur unsharpened edges on the image, and this will prevent detecting unnecessary, unsharpened edges on the image. As a result of this, when clear egdes are obtained, finding contours will be more accurate. The typical use of Gaussing blur is reducing image noise, and detail. When Gaussian blur is applied , there were increase on performance of detection coupon borders. To find the coupon borders, opencv findContours method is applied. This function takes binary images as input. Contour can be explained as curve joining all the continous points along a boundary, having same color or intensity. So findCountors method simply looks these necessities. As input findContours method takes a 8 bit single channel image, non zero pixels are treated as 1's, zero pixels remain 0's, so the image is treated as binary. To create binary images, and increase accuracy of this algorithm, thresholding, canny edge detection and other methods that outputs binary images are used. In our coupon detection code, before finding contours, Canny Edge Detection and OTSU Thresholding is used. In a loop with range of 3, in the first iteration Canny Edge Detection is applied to find edges, the sensivity of this function is middle. On edged image, dilation is applied. Dilation is simply used to make edges wider, to remove holes between edges so that when we run findContours function on this binary image, it wont fail because of this holes. Remainder: findContours method checks for closed shapes. On next iteration, OTSU thresholding is applied to increase accuracy of border detection. FindContours method was looking the color and intensity of the pixels to detect borders, but when user taken image is darker, when where was not enough light to take a clear picture, it is getting hard to detect coupon borders from background. In these cases thresholding is applied to get a binary image as a input for findContours method. The type of thresholding method that is chosen was OTSU. The reason why OTSU method is chosen is that, the algorithm assumes that the image contains two classes of pixels following bi model histogram which means foreground and background pixels, then it calculates the best thresholding value to cut foreground image from the background. To calculate the best threshold value it follows that basic idea: find the threshold that minimizes the weighted within-class variance. So OTSU method is good for thresholding to eliminate the background,which is needed to detect coupon borders.The binary output of this function is input for findContours method. As a result, simple explanation why OTSU thresholding is used is that, to detect coupon borders when edge detection is not enough and coupon borders are indistinguishable, especially on darker images. All these binary images that comes from edge detection and thresholding are performed by findContours method.findContours method gives an output of type vector <vector <Point>>, outer vector represents all the detected shaped, and inner vector holds the points of corners of these shapes.Different data structures may be used. After finding contours, approxPolyDP(in,out,perimeter,bool closed) function is used to approximates shapes to polygons. This function approximates, distorted shapes to decent polygons, the perimeter parameter defines the maximum distance between original curve and its approximation. The last parameter, closed is used to decide whether the polygon is closed or not, since what is needed is border detection, this parameter is set true. See figures below,

Initial contour

Approximated polygon

To find coupon borders, resulting approximated polygons are thresholded by corner count, which means that only polygons which have 4 corners are taken. Resulting quadrangles are thresholded by contour areas, to eliminate small,tiny contours on the image. After these operations program gives a couple of predictions of coupon borders. On 65 samples (does not include solid white background), program detects coupon borders of 61 samples. Success: 61/64. Last thing to do is that determining the best shaped quadrangle by checking angles between sides of quadrangle.

Team evaluation

How well your is team working together? How many meetings did you hold? Are you planning any changes in your cooperation strategy? Which work is completed by which member (in a Gannt chart)?

All members regularly attend all weekly meetings. We were active on OpenProject.
Since new tasks are assigned by our supervisor Asst. Prof. Sinan Kalkan, task distributions are changed this sprint.

The effects of these changes are below:
- Line detection task is assigned to Ögem Çetin. In startup document it was stated that generating detection code (T5) is assigned to both Mesut Yilmaz and Ögem Çetin.
- Recognition of the type of the game (T3) is assigned to Ulas Dalli.
- Also new tasks, namely; ORB algorithm in OpenCV, Binary Descriptor Algorithms other than OpenCV (BRISK), Saving Keypoint Descriptor Information of the Logos in a File(such txt,xml,researched by him as well.

List of new tasks:
NT8 : ORB algorithm in OpenCV
NT9 : Binary Descriptor Algorithms other than ORB in OpenCV (BRISK)
NT10 : Saving Keypoint Descriptor Information of the Logos in a File (such as txt,xml,yaml)
NT11 : 4 Points Perspective Transformation
NT12: Convertion C++ Perspective Transformation Code to Java
NT13: Optimizing Perspective Correction Code (finding Contours Method)
NT14: Checking up on possibility of stealing coupon from photo of coupon.

Task	Assigned Member	1st week	2nd week	3rd week
NT11,NT12	Mesut Yılmaz	■		
NT13	Mesut Yılmaz		■	■
NT14	Mesut Yılmaz			■
T5	Ögem Çetin		■	■
T6	Ögem Çetin	■	■	
NT8	Ulaş Dallı	■		
NT9	Ulaş Dallı		■	
NT10	Ulaş Dallı			■

Backlog Updates

What are your backlog updates?

- NT10 : Saving Keypoint Descriptor Information of the Logos in a File (such as txt,xml,yaml) (Remaining: %70)
- T7 : Extensive testing on coupon samples (Remaining: %100)