Regular Graphical Pattern Detection and Its Applications

by

Shang-Yun (Maggie) Wu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author .....
Department of Electrical Engineering and Computer Science
May 18, 2020

Certified by .....
Randall Davis
Professor
Thesis Co-Supervisor

Certified by .....
Dr. Dana L. Penney
Director of Neuropsychology, Lahey Hospital & Medical Center
Thesis Co-Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Regular Graphical Pattern Detection and Its Applications

by

Shang-Yun (Maggie) Wu

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2020, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

While there is no existing cure to Alzheimer's disease, early detection and intervention can greatly improve patient prognosis. However, early signals can be very subtle changes in behaviors. Our research aims to understand an individual's behaviors through analyzing their gaze patterns. We do this by introducing eye tracking in addition to traditional pen-and-paper tests that measure cognitive status. Traditionally, fiducial markers are added to assist in locating gaze positions with respect to an object in the real world. However, fiducial markers can introduce a distraction and make the test different from its traditional pen-and-paper version. To enable eye tracking without fiducial markers, we present an algorithm that identifies the graphics within the test, allowing us to locate a subject's gaze on the test form using the test alone. It is a novel approach to detecting features in regular graphical patterns despite occlusions.

Thesis Co-Supervisor: Randall Davis
Title: Professor

Thesis Co-Supervisor: Dr. Dana L. Penney
Title: Director of Neuropsychology, Lahey Hospital & Medical Center

Acknowledgments

I would like to express my deepest gratitude to my amazing advisor, Professor Randall Davis of the Human Computer Interactions Lab at CSAIL, and to Dr. Dana L. Penney, Neurology Specialist from Lahey Hospital and Medical Center, for guiding me through this project and being accommodating in the midst of COVID-19. Without their ongoing support and guidance, ranging from academic to personal advice, this project and my level of personal growth would not have been possible. Thank you for absolutely everything and I aspire to one day be as knowledgeable and encouraging as the two of you.

I also wish to acknowledge the following individuals for their inputs and help to my research project. Thank you to Prof. Berthold K.P. Horn for teaching me machine vision concepts that directly impact one major stage of my algorithm. Thank you to Huili Chen, Ph.D. candidate in the Personal Robots Group at the Media Lab, for being an incredible UROP advisor and preparing me for this research experience. Thank you to Yun Boyer for suggesting valuable algorithmic ideas that directly improved my research method. Thank you to Meredith Zhou for reading and editing both my research proposal and thesis numerous times until they were polished and ready for submission.

I would like to additionally thank Elizabeth DeTienne, Karunya Sethuraman, and Sarbari Sarkar for being wonderful lab mates. Thank you for letting me bounce off research ideas, giving me suggestions, and accompanying me throughout this journey.

Finally, I would like to thank my family and friends for always being there for me. Thank you for acting as sanity checks and making sure I stay both healthy and happy in stressful times. I know it would have been a rather difficult journey without your love and support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Cognitive decline is a major health problem worldwide. In particular, Alzheimer's disease - the most common cause of dementia - impacts as many as 1 in 10 individuals over the age of 65 [2]. Those with Alzheimer's typically experience symptoms such as memory loss and time or space confusion, with end-stage Alzheimer's resulting in multiple organ failure and death. The cost of long-term care for individuals with Alzheimer's disease is substantial and, globally, it is estimated to be approximately $605 billion per year [14].

Despite knowing the progression of Alzheimer's disease, researchers have not been able to identify a specific cause. Extensive study has implicated genetic, lifestyle, and environmental factors [15]. Perhaps in part due to this complex and multifaceted nature, a cure for Alzheimer's still remains elusive. Nevertheless, early detection and intervention can greatly improve patient prognosis and there has been much effort devoted into early diagnosis.

For several years, joint research between the Lahey Clinic and MIT has been using hardware that captures both the process and the product of the state-of-the-art tests they developed. The hardware captures both what the subject drew and the process they used to draw them. This allows researchers to capture subtle characteristics in the subject's behaviors during the testing process. Recently, this research has also

incorporated eye tracking, which identifies where the subject is looking and is believed to reflect their thought processes. Our work used an eye tracking headset, instead of an eye tracking chin-rest, because of comfort and the likelihood of capturing the subject's natural behaviors.

An eye tracking headset provides us with the subject's gaze position in the *world camera view*, i.e., the subject's point of view (see Chapter 2.1). We also need to locate the subject's gaze on our test form. Locating the test form is typically done using fiducial markers, which are graphical patterns (Fig. 1-1) that can be easily recognized. A fiducial marker is placed at each of the four corners of the test so that, in most cases, at least two of them are visible in the world camera view.

While the use of fiducial markers vastly simplifies the vision task, a test with fiducial markers is inconsistent with the traditional version administered and may be distracting for the subject. In order to keep the data collected when using eye trackers consistent with those from the traditional pen-and-paper tests, it is important that the two tests are the same.

Figure 1-1: Sample Fiducial Markers

The goal of this research is thus to detect and locate the test form in the world camera view without using fiducial markers. We aim to achieve accuracy within half a degree of the visual angle even when the test form is partially occluded. We believe the algorithm developed to locate the test form may also contribute to object detection in the field of computer vision.

Note that since all of our tracking data is recorded (including videos from all three cameras), we do not need real-time processing. This allows us to explore a wider range of approaches.

This thesis proceeds as follows. Chapter 1 provides motivation and background for our study and the success and limitations of our research outcome. Chapter 2 elaborates on eye tracking and the test forms. Chapter 3 references related studies in

the field of cognitive science and computer vision to illustrate the difference between past studies and our research work. Chapter 4 explains our algorithm in detail and how it is used to achieve our goal. Chapter 5 details the evaluation metrics and presents results obtained from our algorithm. Finally, Chapter 6 concludes with the effectiveness and limitations of our algorithm as well as future work that can be extended from this research.

1.2 Attempted Approaches

We tried several different methods to detect and locate the test form. Here, we review briefly several that did not work and explain why not.

1.2.1 Why Not Object Tracking?

One plausible solution is object tracking. Our test form is always readily available in front of the subject at the start of the study, giving us a clear image of the test form in the world camera view and making it easy to identify and create a rectangular box around it. If we were to use object tracking, we would start a recording, pause it at the start, click and drag to create a rectangular box, i.e., the object tracker, around our test form, and let the object tracker follow it throughout the video.

While this is a plausible solution, object trackers are generally prone to shifts as a result of occlusions[1]. When an object to be tracked is occluded, its object tracker loses information about it. As a result, the object tracker typically can no longer capture the location of its object and may instead start following the source of occlusion (e.g., a moving hand). Even with the help of multiple object trackers situated at various locations of the test form, we cannot guarantee that all of these locations will be visible during a testing session. In fact, we found that object trackers return the accurate location of the test form less than 50% of the time.

[1] An occlusion occurs when part or all of the object is hidden from the vantage of observation

1.2.2 Why Not Scale-Invariant Feature Transform (SIFT)?

SIFT is an algorithm to describe and detect objects in an image [11]. It does so by extracting and identifying keypoints, which are points that lie in high-contrast regions (e.g., an edge) of an object. The specific distribution and pattern of keypoints, known as a feature vector, usually uniquely defines the object of interest. Furthermore, these feature vectors are robust to scale, noise and illumination, making SIFT a popular technique for object detection tasks.

SIFT works well for identifying objects in natural images because individual objects in the wild have distinct patterns, resulting in unique feature vectors. However, the graphical layout of our test has a regularity that makes the local neighborhoods similar to one another, causing significant problems for SIFT.

1.2.3 Why Not Boundary Detection?

Boundary detection is a process to detect and localize object boundaries, which we thought might be useful in identifying the contours of the test form. In fact, we did find this to be a powerful technique when the test form is only slightly occluded. However, because the technique relies on finding the four sides of the rectangular test form, when any one of them is heavily occluded, this technique loses its reliability.

1.3 Successes and Limitations of Our Algorithm

Our algorithm, the Regular Graphical Pattern Detection Algorithm, is designed to identify pieces of our test form as fiducial markers. It first removes distortions[2] caused by lens and perspective. Then, it identifies patterns within the graphics of the test, which in turn tell us where the test form is located within the world camera view. Knowing the gaze location, as provided by the eye tracker, and the test form location in the world camera view, our algorithm computes a transform that helps identify exactly where the subject is looking on the test form.

[2]In geometric optics, distortion is a deviation from rectilinear projection - a projection in which straight lines in a scene remain straight in an image.

All video frames were processed using a 13-inch MacBook Pro with 2.8 GHz Quad-Core Intel Core i7 processor. On average, our algorithm took 10 seconds to compute a suitable transform for a frame taken at one-thirtieth of a second. The amount of time a subject took to finish a test ranged from 15 to 30 seconds. Consequently, it took approximately 2-3 hours to completely process each video.

Clearly, the processing time would be considerably shorter with a more powerful machine. As noted earlier, we do not need to process the video in real-time. All of the videos can be processed offline, making the processing time far less significant.

1.3.1 Successes

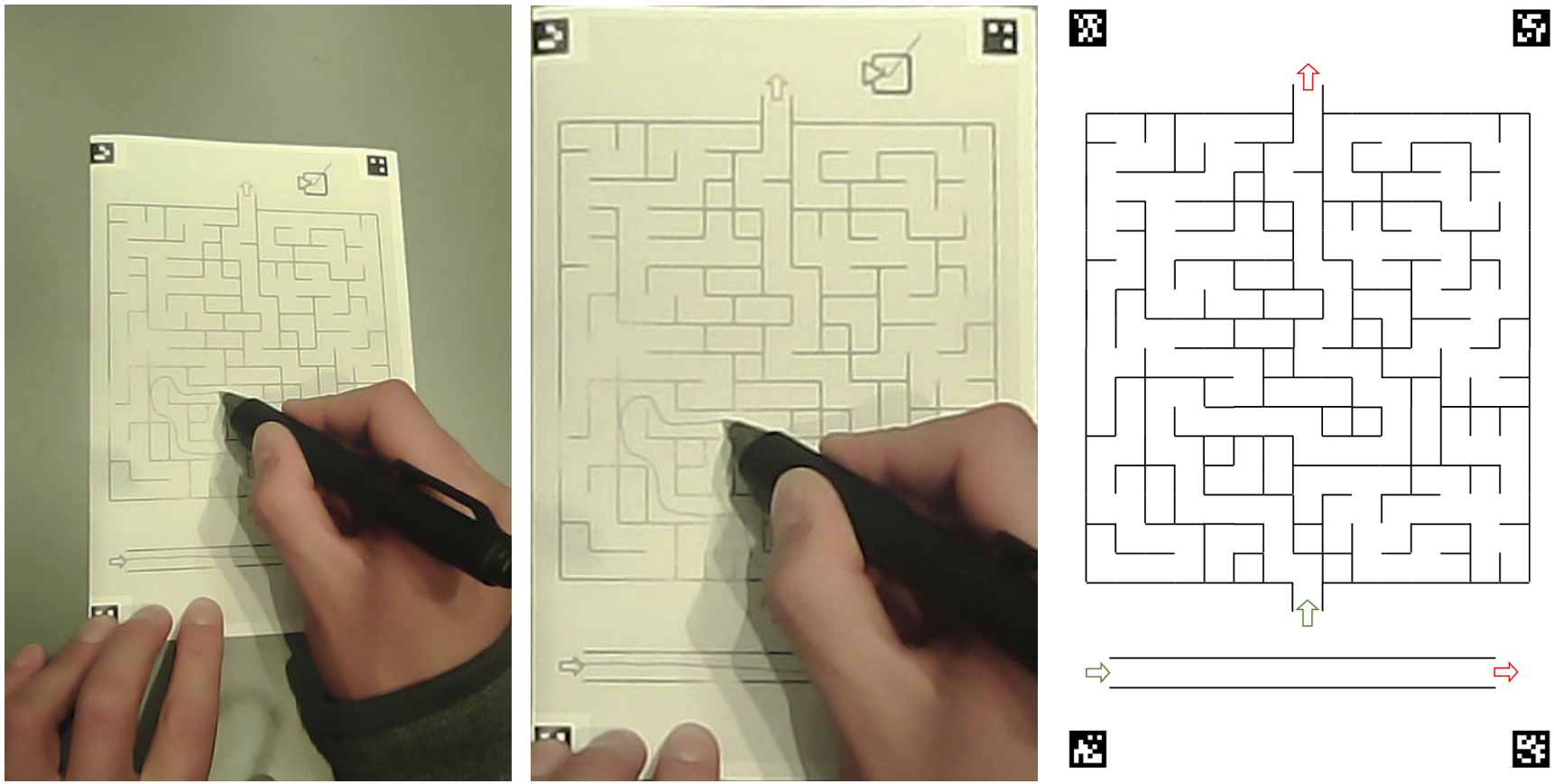
We define processing of a frame to be a success if we transform the image so that the transformed image matches the ground truth template with a considerable accuracy (i.e., an error within 50% of the width of the pathway). Chapter 5.1 provides specific evaluation metrics and Fig. 1-2 shows an example of successful processing.

Our algorithm was able to successfully process most frames across videos with different illumination and backgrounds. On average, it achieved a hit rate of 89.75%, a sensitivity of 90.00%, and an average median distance of 1.27mm across four subject videos[3]. Our algorithm worked well under both real-world and purposely stressful conditions. These stress tests capture some of the most extreme behaviors clinicians are likely to observe in patients during the study, including rotation, lean, and shadow. Specific results can be found in Chapter 5.2.

Because our algorithm is designed to dynamically compute the appropriate values for the parameters of the edge and corner detectors (as explained in Chapter 4), it does not need users to hand-pick parameter values for different videos. Our algorithm can thus be applied to different video recordings with little fine-tuning. We also expect the algorithm to generalize to various types of tests with minor modifications[4].

[3] As discussed in Chapter 5.1, hit rate is the percentage of frames processed successfully; sensitivity is the percentage of corners correctly identified; median distance is the median of the distances between all transformed corners and their ground truth locations.

[4] All figures shown in this thesis have fiducial markers. We worked off of this version of the form in order to compare the gaze positions obtained using our algorithm vs. using Pupil Labs' software.



(a) Original Image (b) Transformed Image (c) Ground Truth Template

Figure 1-2: An Example of Successful Processing

1.3.2 Limitations

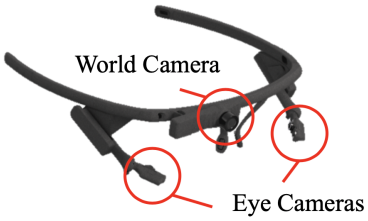
One challenge for our algorithm is frames that are blurred, typically caused by a sudden shift in the head position. Because we rely on good feature detection, when images are blurred and features are hard to detect, our algorithm struggles to perform well. Rapid head movements are, however, not the only cause of video frames for which we cannot compute appropriate transforms. There are a number of other situations that cause this issue but we still do not entirely understand why this happens. As a consequence, as explained in Chapter 4.6, we developed a variety of techniques to deal with frames for which the appropriate transform cannot be computed from that frame alone.

Chapter 2

Technical Background

2.1 Eye Tracking

The eye tracker (Fig. 2-1a) used in our study was developed by Pupil Labs. It has 3 cameras with video recording capabilities. Two of the cameras, called the eye cameras, focus on the left and right eyes, respectively. They estimate the positions of the pupils, which indicate gaze direction. The third camera, called the world camera, is used to record the subject's point of view.



(a) Eye Tracker

(b) Calibration Target

Figure 2-1: Eye Tracking Equipment

At the start of a study, we adjust eye cameras so they capture both pupils entirely and focus the world camera so it approximately reflects the subject's point of view. A calibration process is next in which we ask the subject to follow a target (Fig. 2-1b) with their eyes alone while it is moved around within the world camera view. This target is designed to be easily recognized and is located by Pupil Labs' software. As a result, these three cameras can, together, identify where the subject is looking.

2.2 Cognitive Tests

2.2.1 Test Types

Two different tests, the maze test and the symbol-digit test, are of interest here. Because our algorithm is currently designed to work for the maze test, this thesis provides figures specifically for the maze test. However, some figures and descriptions of the symbol-digit tests are in Appx. A-1 as we expect a similar algorithm will generalize to the symbol-digit test.

The maze test is a 16×15 grid-like structure where the width of any pathway within the maze is constant, approximately 8mm when printed. This test requires a subject to find a solution path from the start (i.e., the bottom arrow) to the goal (i.e., the top arrow).

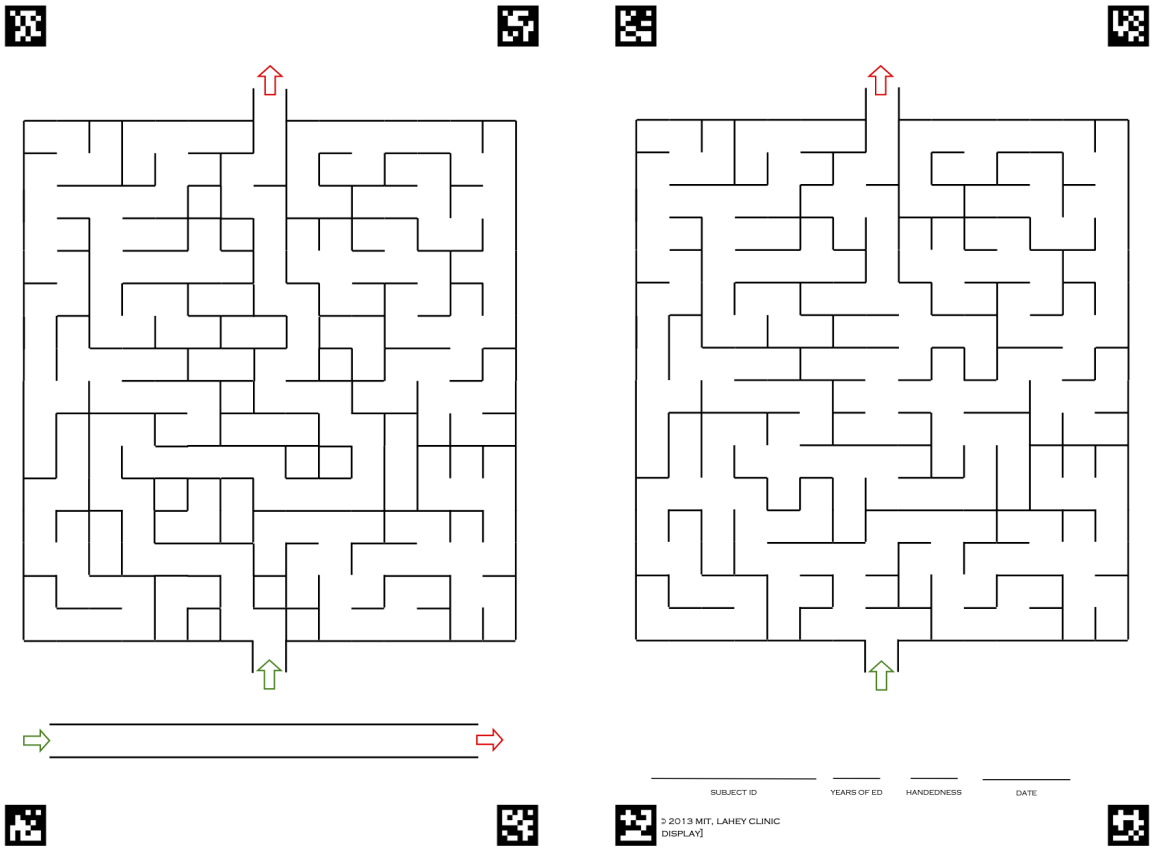
2.2.2 Test Conditions

The maze test has two conditions, low cognitive load (LCL, Fig. 2-2a) and high cognitive load (HCL, Fig. 2-2b). Each subject completes both conditions twice in a study. The subject is presented with each condition in turn and cannot move back and forth between them.

The LCL test is a maze with no choice points. The subject simply follows the path from beginning to end.

The HCL test consists of a maze with choice points, i.e., multiple paths. Subjects have to find the path that leads to the exit.

Unknown to the subjects, the solutions to both the LCL and the HCL tests are identical. Having identical solutions means the subject is presented with two tasks that require identical physical demands but different cognitive loads. The subjects thus act as their own controls, enabling the test to distinguish physical and cognitive issues in performance.



(a) The LCL Test

(b) The HCL Test

Figure 2-2: The Maze Tests

Chapter 3

Related Work

There are two areas of related research: digital cognitive assessments and object detection with occlusion.

3.1 Digital Cognitive Assessments

Other work has compared patient performance on digital cognitive assessments to those on corresponding pen-and-paper tests [1, 6, 19]. For the most part, studies concluded that additional research is necessary before their respective technologies can be used to replace traditional pen-and-paper tests. This arises because of the small sample sizes used in these studies. It is therefore difficult to say whether or not the results will generalize to a larger population. However, initial pilot experiments showed comparable and promising results between the two versions of the assessments [6, 19].

These relate to our work as they demonstrate the effect on patient performance with digital technology as the input. In our case, the additional technology introduced is the eye tracking headset. Future work should compare patient performance with and without the eye tracking headset. This can then help isolate the effect of the headset, a potential source of distraction.

3.2 Object Detection with Occlusions

Occlusion is common in images, posing a challenge to object detection. While SIFT is a robust technique for detecting texture-rich objects, it struggles to detect those with uniform or repeating patterns, such as the maze and the symbol-digit test. In addition, the use of arbitrary viewpoints and the presence of occlusion further adds to the difficulty of the problem.

One model proposed by Hsiao et al. addresses these common difficulties - textureless objects, arbitrary viewpoints, and occlusions [5]. The model describes the 3D interactions between objects, which indirectly explains how an object occludes another in an arbitrary viewpoint. As a result, this model can be used to describe how occlusions affect the particular object we are trying to detect. By training the model to detect a particular object under various types of occlusion, it can achieve an average detection precision of 77% when occlusion was slight ($\leq$35%). However, the model struggled in detection performance when there was heavy occlusion ($\geq$35%).

Given that our research focuses on pattern recognition under occlusion, an understanding of the occlusion itself is not necessary. We do, however, require an accuracy beyond what this model can achieve.

In the past decade, there has been increased interest in using machine learning models to automatically construct a bounding box around the object of interest. These models have been successful in identifying pedestrians and vehicles [8, 18]. Some even gained success in achieving fast real-time detection [17]. However, they are mostly designed to work well in a single view, which does not satisfy our research needs; using the eye tracking headset, our participants are free to move their heads, leading to multiple views. In addition, these models often sacrifice some accuracy for efficiency, but our research instead favors accuracy over efficiency.

For both occlusion understanding and many machine learning models, a bounding box is created to capture the object of interest. However, a bounding box for object detection does not describe the location of the object at the pixel level. Boundary detection with occlusion has been proposed in an attempt to resolve this concern [7].

For simple geometric shapes, the description for an object's boundary is as accurate as 95% when 60% of the object is occluded. This method, however, suffers when there is very little or heavy occlusion, which is a critical shortcoming for detection that requires high accuracy under various levels of occlusion.

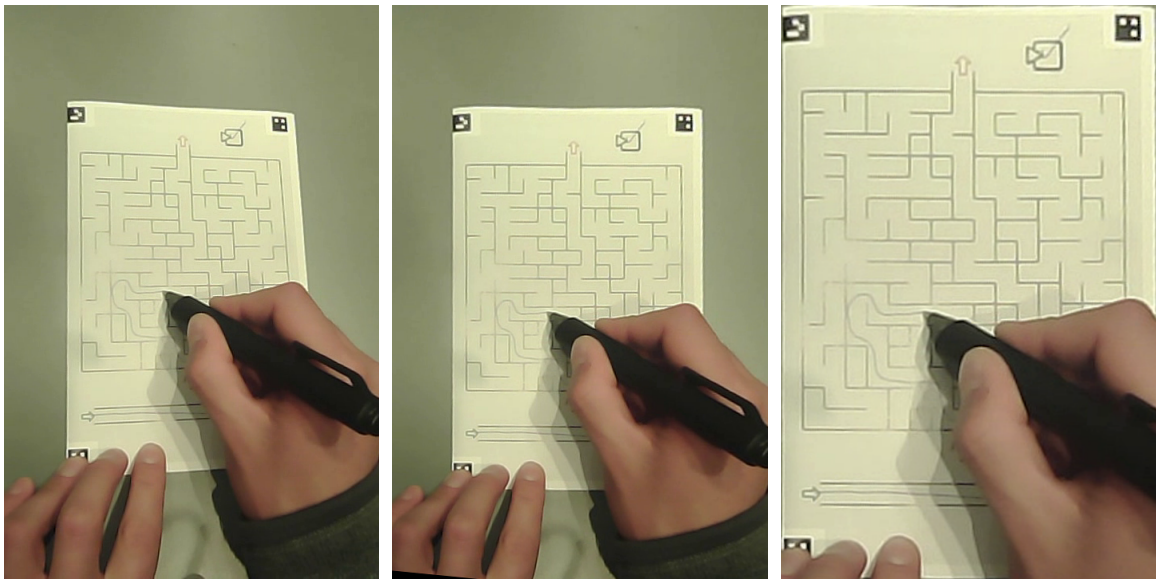
Many object detection models and algorithms have, thus far, focused on a combination of accuracy, efficiency, and generalizability. As noted above, efficiency is not a pressing concern to our research because our videos can be processed offline. In regard to generalizability, cognitive tests are often designed to have specific patterns in order to effectively evaluate an individual's mental state. These patterns, albeit repetitive and similar, hold valuable details that we can leverage for detection. That is, we believe an algorithm that works for one type of test can generalize to other tests in the same field. Our work, as a result, focuses on understanding and detecting regular graphical patterns with high accuracy.

Chapter 4

Research Approaches

4.1 Overview

To effectively detect, identify, and localize the test form in a frame of the video, we must transform the frame from the *world camera view* (Fig. 4-1a) to the *rectified view* (Fig. 4-1b), then further transform to the *standardized view* (Fig. 4-1c). These two major transforms and their intermediate steps make up the Regular Graphical Pattern Detection Algorithm.



(a) World Camera View (b) Rectified View (c) Standardized View

Figure 4-1: Different Views in the Process

Stage one of the algorithm rectifies a frame from the world camera view (Fig. 4-1a) to the rectified view (Fig. 4-1b) using vanishing points. This eliminates perspective distortion, making it easier to detect regular patterns. We call this stage the Image Rectification.

The maze is designed to have all pathways have the same width. This is evident in the rectified view (Fig. 4-1b). We define the *unit* to be the width of the pathways; we also define a corner to be the endpoint of any line or where two lines in the maze intersect. We can see that each corner in our particular maze has a unique pattern of other corners that lie within 2 units of it[1]. Stage two of the algorithm uses these *neighborhood patterns* to describe pieces of the test form. We call this stage the Regional Pattern Analysis.

With some neighborhood patterns identified, stage three, the last stage, of the algorithm computes the homography that transforms a frame from the rectified view (Fig. 4-1b) to the standardized view (Fig. 4-1c). Again, the standardized view is the final view our algorithm aims to achieve. We call this stage the Homography Transformation.

When we cannot find the vanishing points, the unit, or the homography at their respective stages for a given frame, they are assumed to be the same as those of its previous frame. This is a fallback technique we call *repetition*. We use repetition based on the assumption that two consecutive frames taken at the normal video rate, i.e., one-thirtieth of a second apart, are likely to have parameters very similar to one another. The repeated parameters are then used to process the current frame of interest in the aforementioned way.

4.2 Image Rectification

To transform an image from the world camera view (Fig. 4-1a) to the rectified view (Fig. 4-1b), the algorithm masks the image to remove background, detects edges and lines, and computes vanishing points to be used for rectifying the image.

[1]We will explore the cases where not every corner has a unique pattern around it as future work.

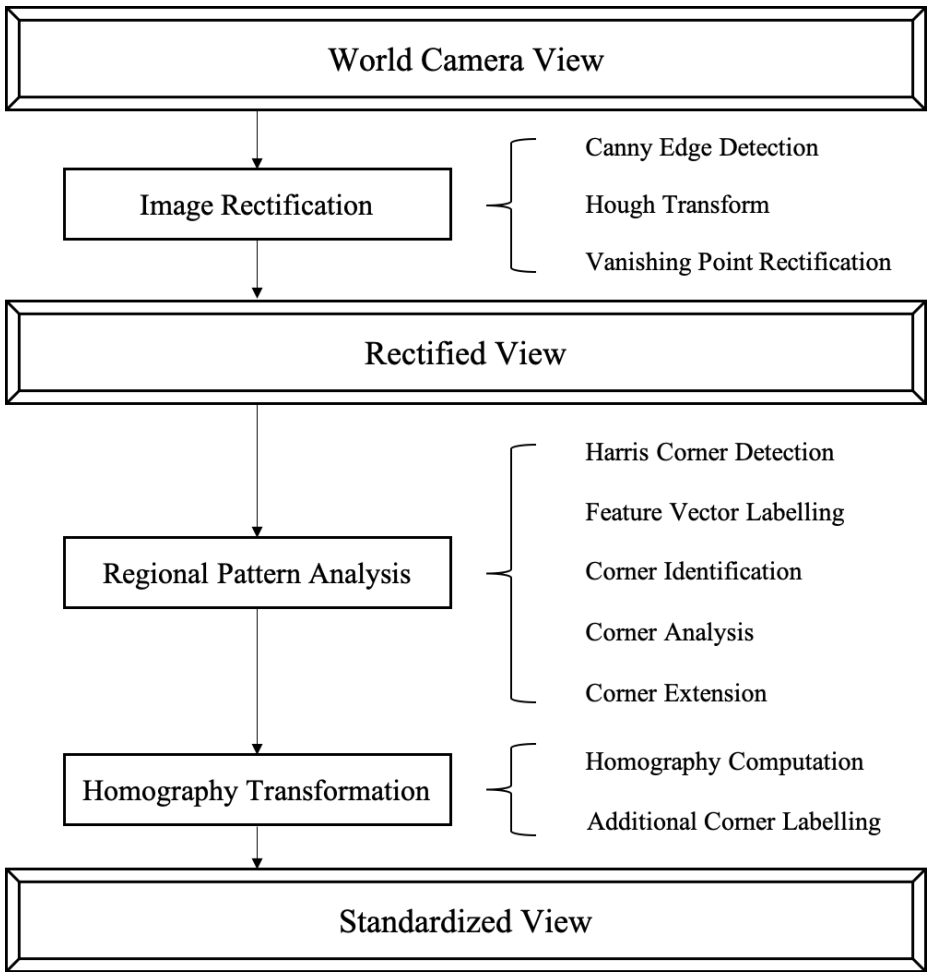
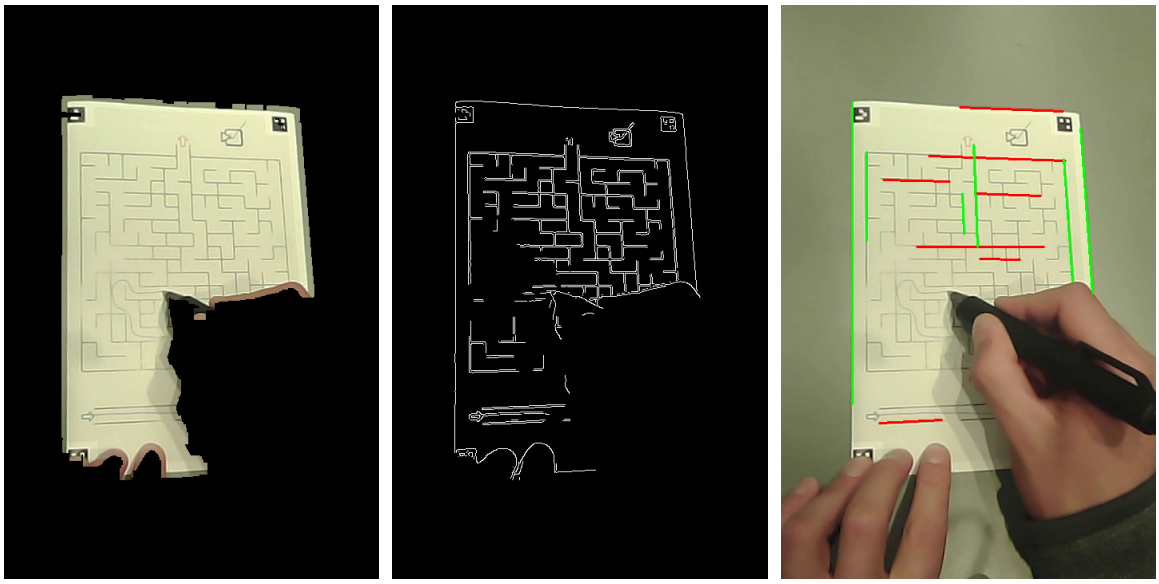


Figure 4-2: Workflow Summary. The Regular Graphical Pattern Detection Algorithm includes 3 views (2 transforms), 3 stages, and 10 sub-stages.

4.2.1 Hue-Saturation-Value (HSV) Mask

To avoid visual background noise, subjects take the test while seated at a dark, monochromatic desk. This makes our white test form stand out. Determining the range of HSV values suitable for a recording is done via a brief manual process. At the start of a video, the user clicks on the four corners of the form to allow our system to identify their HSV values. Among these, the minimum and the maximum define the range (typically between [20, 30, 150] and [50, 60, 240]). Afterwards, we apply a HSV mask, an image filter that removes pixels with HSV values outside that HSV range, to eliminate most of the background in the world camera view. This creates the *masked image* (Fig. 4-3a) and limits our feature detection to within the test form.



(a) Masked Image (b) Edge Image (c) Hough Lines

Figure 4-3: The Image Rectification Intermediate Results

Because the lighting within an experiment room is relatively consistent over the course of a test (about 10 minutes), a single HSV mask is sufficient.

The algorithm works in the HSV space, rather than the RGB space, because HSV can capture changes in lighting due to shadowing. It does this by separating image intensity from color information. Color information is captured by hue while image intensity, which contains brightness information, is captured by saturation and value. By modifying mask thresholds with respect to saturation and value, we can effectively capture relevant pixels in all of our frames.

4.2.2 Edge Detection

To use the graphics of the test itself as fiducial markers, we need to detect features, i.e., identifiable patterns, within the test. To do this in turn, the algorithm recognizes features in the masked image that make up identifiable patterns. In particular, it uses the *Canny edge detector* and the *Hough transform* to detect edge pixels and lines.

The Canny edge detector is used to detect edge pixels in an image [3]. Even though developed in 1986, it is still one of the most effective and commonly used edge detectors today.

In general terms, it blurs an image with a Gaussian filter, computes image intensity gradients, and selects edge pixels using non-maximum suppression and hysteresis thresholding. Everything considered, it takes in a masked image and outputs an *edge image* (Fig. 4-3b) representing edge and non-edge pixels.

Non-maximum suppression is the key to finding thin edges. An edge pixel is selected if it has the maximum image intensity gradient among those of its neighboring pixels in the direction of the image intensity gradient. This step in the algorithm effectively suppresses many false positive edges and preserves only pixels with sharp image intensity changes.

Hysteresis thresholding filters edge pixels with respect to a *minimum edge value* and a *maximum edge value* (Fig. 4-4). An edge pixel with an image intensity gradient less than the minimum edge value is discarded and one with an image intensity gradient greater than the maximum edge value is a true edge. Those with image intensity gradients in between are accepted if they are connected to a true edge.

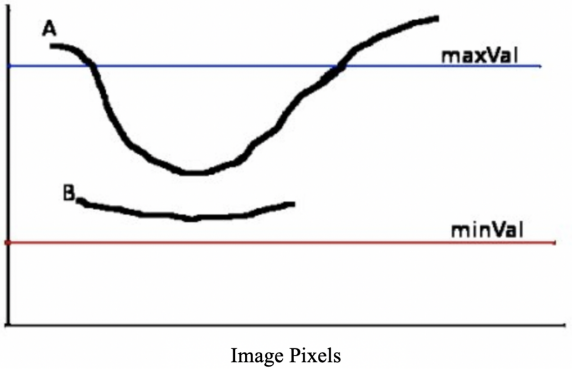


Figure 4-4: Hysteresis Thresholding. Pixels in A are true edges because they are either above the maximum edge value or connected to those who are above the maximum edge value. Pixels in B are not true edges.

The choice of edge values, however, depends on the application. Lower edge values result in more edge pixels detected and higher edge values result in fewer. For our purposes, we empirically chose small edge values because we were more concerned with false negatives (edges missed) than with false positives.

4.2.3 Line Detection

The Hough transform is a technique that efficiently and effectively detects common geometrical shapes, specifically lines in our edge image [16]. The underlying principle of the Hough transform is a voting system for lines. For every edge pixel detected by the Canny edge detector, we use the Hough transform to create multiple line candidates at different angles. If a candidate intersects with many other edge pixels, meaning these edge pixels all fall under the same line, the candidate is likely a true line and is accepted.

The parameters to the Hough transform include the *threshold*, the *minimum line length*, and the *maximum line gap*. The threshold value is the minimum number of edge pixels a line must pass through. A high threshold generally results in fewer lines detected and a low threshold results in the opposite. The minimum line length, as is evident from its name, is a threshold for the lengths of the lines detected. The length of a line is defined by edge pixels on the two ends of the line. The maximum line gap is the distance allowed between a line and an accepted edge pixel. Because edge pixels detected may not lie perfectly on the line, it is necessary to allow some gaps in order to include edge pixels that fall close enough to a line. This relaxation is especially important for real-world images.

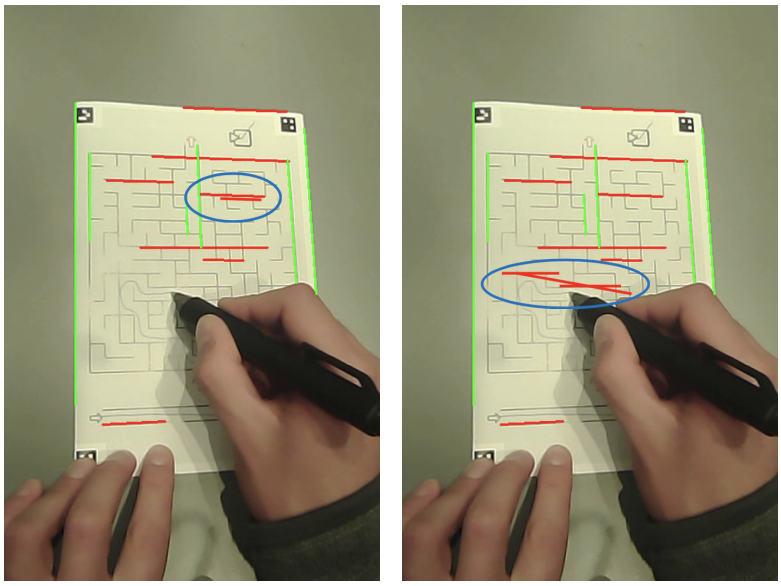
For our purposes, parameters for the Hough transform are empirically chosen so that a line is detected only when a large number of edge pixels lie on the same line with a small maximum line gap. To further ensure that enough lines are detected to compute vanishing points later in this stage, both threshold and maximum line gap are adjusted dynamically. From their respective default values, the algorithm lowers the threshold and increases the maximum line gap until it has found 10 horizontal lines and 10 vertical lines, knowing that there are multiple such lines in the test form.

There are two tests - the closeness test and the overlap test - designed to remove lines that don't follow the maze grid. For simplicity, we call these the *non-grid lines* and the ones that follow the maze grid the *grid lines*. The non-grid lines are detected by the Hough transform when enough edge pixels are found diagonally. This

phenomenon is usually present when the maximum line gap becomes so large that edge pixels across multiple lines on the maze grid can be connected to constitute a detected line.

The closeness test removes detected lines that are too close to each other. Because there are two sides to a line on the maze grid, sometimes both sides are detected as separate lines in the image (Fig. 4-5a). To ensure that lines detected correspond to different lines on the maze grid, we remove the shorter of any two near-parallel lines that are less than 5 pixels apart.

The overlap test checks that no detected lines intersect with each other. While these lines, when extended very far, should theoretically intersect at a vanishing point, they should not, in pair-wise, intersect at a point anywhere close by. In other words, a line that intersects with multiple lines within the maze is likely a non-grid line (Fig. 4-5b). Our algorithm iteratively removes such lines until no more are left.



(a) The Closeness Test (b) The Overlap Test

Figure 4-5: Non-grid Lines Removed by The Closeness & The Overlap Test

After filtering the line candidates with these two tests, we are left with mostly, if not only, grid lines. We call these the Hough lines. To demonstrate, we overlay the horizontal Hough lines (red) and the vertical Hough lines (green) with the world camera view (Fig. 4-3c).

4.2.4 Vanishing Points Rectification

We know that the vertical and the horizontal Hough lines are parallel in 3D. The world camera view of the form makes them look nonparallel and appear to meet at two vanishing points, one for vertical lines and one for horizontal lines. Our goal is to undo this perspective distortion caused by the 2D view and return to the view of the form where the parallel lines are once again parallel, i.e., the rectified view.

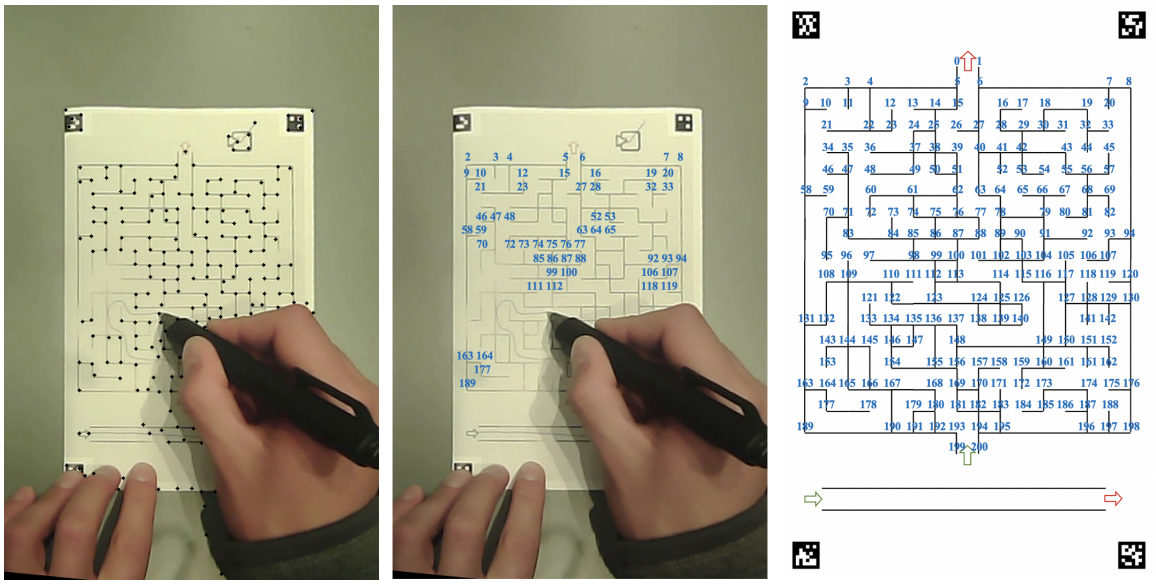
To do so, the algorithm first computes the horizontal and the vertical vanishing point. Consider the set of vertical Hough lines for the moment. Given two lines within this set, it is easy to compute their vanishing point, i.e., their intersection. However, if there are more than two lines in the set, it becomes necessary to perform least-squares to find a *best-fit* vanishing point, a point that minimizes the sum of squared distance to all the lines used to compute it.

Using the horizontal and the vertical vanishing point, the algorithm transforms the image from the world camera view to the rectified view using parallel projection, a method to transform an image from a 3D to a 2D view by projecting the image onto a fixed plane. Here, the vanishing points tell us where the fixed plane is so the projection can be done. Finally, bicubic interpolation[2] is used to fill in any gaps, i.e., non-integer pixel values as a result of any transformation, in the rectified view.

4.3 Regional Pattern Analysis

Working from the rectified view (Fig. 4-1b), we want to detect, then identify, corners in the maze. Our algorithm finds corners in the maze, labels each one using its unique neighborhood patterns, i.e., feature vectors, and determines which corner in the ground truth it matches to. These identified corners are used to compute the homography transformation.

[2]In mathematics, bicubic interpolation is an extension of cubic interpolation, a third-degree polynomial interpolation, for interpolating data points on a two-dimensional regular grid.



(a) Harris Corners (b) Identified Corners (c) Ground Truth Corners

Figure 4-6: The Regional Pattern Analysis Intermediate Results

4.3.1 Corner Detection

The corner detection process includes detecting corners then filtering them.

A corner detector operates on the image intensity gradient. Our algorithm uses the *Harris corner detector* to detect corners in the rectified view. It is an improvement to the traditional *Moravec corner detector*. We first discuss the principles behind the Moravec corner detector as it provides a baseline for how the Harris corner detector works.

The Moravec corner detector identifies corner pixels by looking at the image intensity gradient in patches around the pixel of interest. A corner is detected if neighboring patches are dissimilar where similarity is measured by the sum of squared difference (SSD) of the image intensity gradients. That is, a corner is a pixel where there is a large SSD between the patch at that point and the patch centered at its neighbor [13].

While the Moravec corner detector is effective, its lack of isotropy[3] is a major drawback. The Harris corner detector is an improvement to the Moravec corner detector because it considers the image intensity gradient in the direction of the

[3]Isotropy refers to uniformity in all directions and orientations

maximum image intensity gradient, thereby preserving isotropy. It computes the eigenvalues of the image intensity gradient matrix. Because eigenvalues directly reflect the magnitude of the image intensity gradient, large eigenvalues indicate a corner pixel is present.

The Harris corner detector has only a single free parameter k, which is related to the threshold for the magnitude of the eigenvalues. The larger k is, the more evident the corner must be in order to be identified. We set this to be 0.5, which has been empirically tested to work well.

To control the corner detection process and selectively accept the corners returned by the Harris corner detector, we set three additional parameters. These parameters are the *maximum number of corners*, the *quality level*, and the *minimum distance*.

The maximum number of corners, n, constrains the number of corners that can possibly be returned as the output of the corner detection process. If there are fewer corners detected than the constraint allows for, all corners are returned. Otherwise, the top n corners are returned.

As Fig. 4-3a shows, our HSV mask blocks everything in the frame other than the test form (and pixels with HSV values similar to the test form). As a consequence, a subject's hand and the digitized pen, which are common sources of occlusion, are typically masked out. Therefore, the size of the mask reflects the degree of occlusion. The bigger the occlusion, the fewer visibly available corners there are. Knowing this, the algorithm adjusts the maximum number of corners to be returned by the Harris corner detector according to the size of the mask.

The quality level (QL), defined as a fraction of the maximum image intensity gradient (E_{max}), characterizes the minimally accepted quality of the returned corners. It thresholds on the image intensity gradient (E) to select only candidates whose image intensity gradients are greater than the quality threshold (QT). Note that the acceptance threshold depends on the maximum image intensity gradient of the corners in the image. Hence, the image itself, rather than the user, tells the algorithm what threshold to use.

$$\text{Accept (Corner)} = \begin{cases} \text{True}, & \text{if } E \geq QL \times E_{max} = QT \\ \text{False}, & \text{otherwise} \end{cases}$$

We are overall less concerned about false positive corners than false negative ones given the rigorous analysis further on in our workflow. As a result, the quality level parameter can be small; we have empirically chosen it to be 10^{-4}.

Finally, the minimum distance, which we called the unit distance, ensures that no two corners can be closer than this threshold. This in turn reduces the probability of double-counting corners found in a blurred image.

The minimum distance parameter should be slightly less than the width of the pathways, i.e., the unit, in the maze in order to capture the corners without detecting noise in the pathway. The user first indicates, by eyeballing, the unit size at the start of each video. This user-supplied distance is called the *backup unit* and is usually in the range of 10 to 15 pixels. Then, for each frame in the video, the algorithm produces its own estimate of the unit size from the distances between the Hough lines. Both the backup unit and the estimated unit are used individually to carry out the remaining Regional Pattern Analysis. Whichever unit yields more identified corners at the end of this stage is chosen.

The corner detection process uses the parameters described above and returns a set of detected corners in the rectified view. We call these the Harris corners and visualize them by marking them in the rectified view (Fig. 4-6a).

4.3.2 Feature Vector Labelling

Each corner in our particular maze has a unique neighborhood pattern. We take advantage of this to define feature vectors to describe these neighborhoods.

To explain the basic idea behind the feature vectors, we start with a ground truth image of the corners in the maze (Fig. 4-6c). For our maze, each neighborhood pattern is unique when the pattern includes corners that lie within 2 units of the corner of

interest. We first create 8-bit feature vectors to describe the 1-unit neighborhood pattern of each corner. Then, we create 64-bit feature vectors to describe the 2-unit neighborhood pattern, building on the 8-bit feature vectors.

An 8-bit feature vector of a corner identifies the presence of its neighboring corners. Each bit corresponds to one of 8 locations 1-unit vertically, horizontally, and diagonally away from the corner of interest. Starting from the neighboring location 1-unit vertically above and moving clockwise, we assign a 1 if there exists a corner in close proximity to the neighboring location; otherwise, we assign a 0 (Fig. 4-7). Note that close proximity, instead of exact location, is used in order to account for the slight variations in the Harris corners that are found in the images; sometimes, the Harris corners may slightly off-set from the exact intersections or endpoints of the lines in the maze.

We then create the 64-bit feature vector of every corner by concatenating its eight neighboring 8-bit feature vectors (Fig. 4-7). If no corner is present at a particular neighboring location, the corresponding 8-bit is replaced with 8 dashes as fillers to maintain a constant feature vector size of 64 elements.

Our algorithm automatically creates the feature vectors from the Harris corners. It needs to compute new feature vectors for every frame because the feature vectors depend on the Harris corners, which depend on the rectified view. Luckily, the feature vectors for the ground truth corners are always the same for a given maze.

While a 64-bit feature vector contains repetitive information, the structure is used for its implementation simplicity. It also allows us to easily track sources of error.

4.3.3 Corner Identification

Recall that we create feature vectors for both the ground truth corners in the maze template and the Harris corners in the rectified view. To perform corner identification, we compare the 64-bit feature vector from each ground truth corner to the 64-bit feature vector from every Harris corner. Each ground truth corner sufficiently similar to a Harris corner is considered a potential candidate for that Harris corner.

There are two metrics for evaluating similarity: bit difference and corner difference.

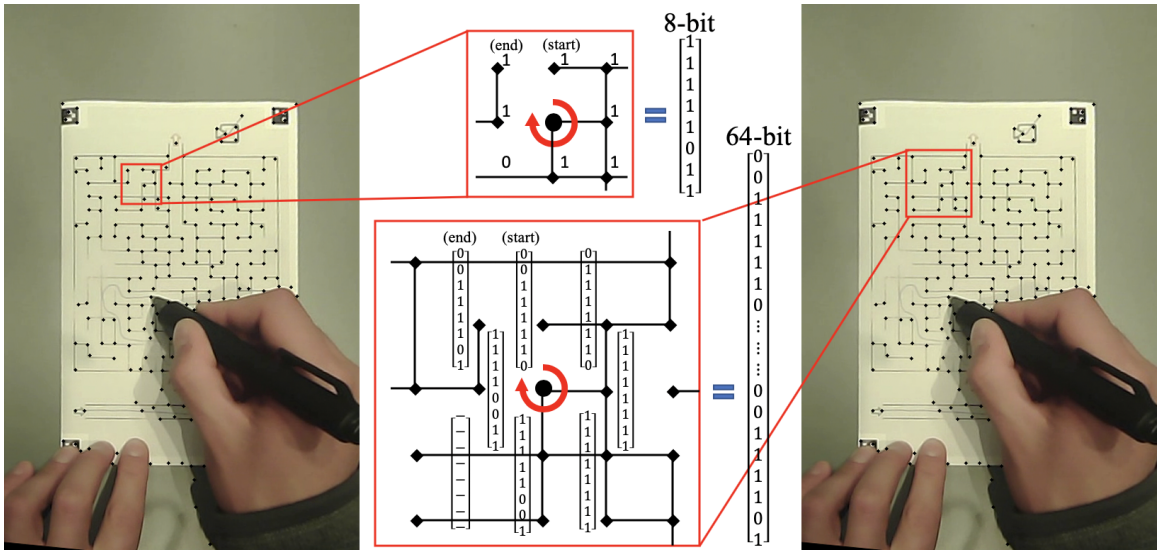


Figure 4-7: 8-bit & 64-bit Feature Vectors

Here, the bit difference is the number of positions in which the two vectors differ. The corner difference, by contrast, considers only differences between a 1 and a dash or a 0 and a dash. Fig. 4-8 shows an example demonstrating these two metrics.

```
[00110011 10110011 00111011 00110011 00110011 00110011 00110011 00110011]
 |        ||       |                 ||||||||
[01110011 01110011 00110011 00110011 -------- 00110011 00110011 00110011]

[ *       **       *                ########                            ]

Bit Difference = 4 + 8 = 12
Corner Difference = 8
```

Figure 4-8: Sample Bit and Corner Differences

Using a combination of bit difference and corner difference yields better identification accuracy by reducing the false negative rate. A simple bit difference can overstate the difference in a potential match. If there is a single corner that has been missed in the image, the bit difference metric will report a bit difference of 8 when in fact all of these 8 differences are the result of a single corner that has been missed. Taking account of both the bit difference and the corner difference helps to deal with this situation.

For each Harris corner, we then sort candidate matches from best to least-matching according to Alg. 1.

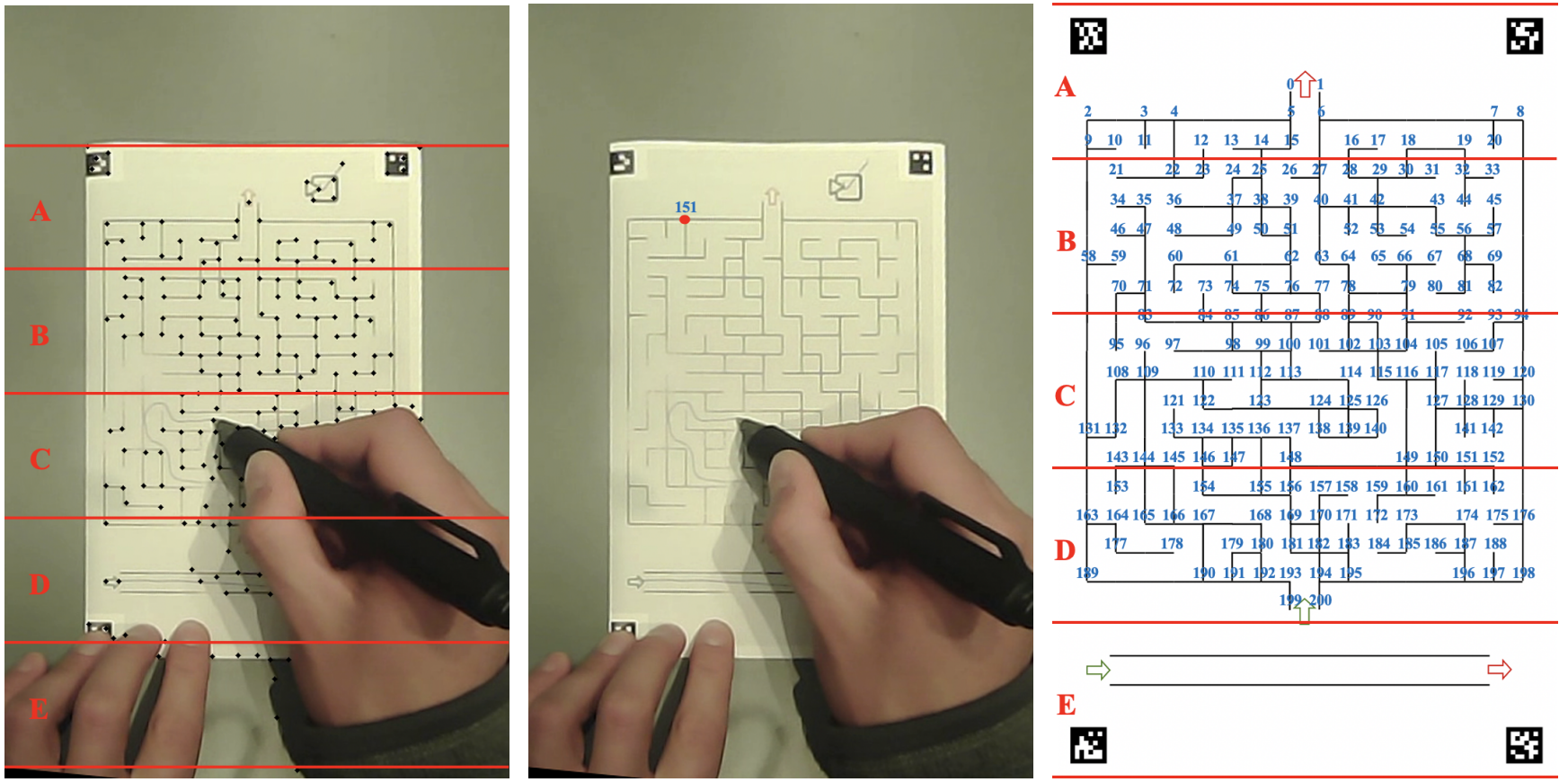
Algorithm 1: SortCandidate

```
initialize bestMatch[], oneMiss[], twoMiss[];
initialize index = 0;
initialize bitDiff, cornerDiff;

// go through all ground truth corner candidates
while index < len(Candidate) do
    bitDiff = Candidate_bit[index];
    cornerDiff = Candidate_corner[index];
    if (bitDiff <= 8) then
        // ground truth corner candidate matches well
        bestMatch.insert(Candidate[index]);
    else if (bitDiff <= 12 && cornerDiff <= 8) then
        // ground truth corner candidate differs by 1 corner
        oneMiss.insert(Candidate[index]);
    else if (bitDiff <= 18 && cornerDiff <= 16) then
        // ground truth corner candidate differs by 2 corners
        twoMiss.insert(Candidate[index]);
    else
        pass;
    end
    index ++;
end

bestMatch.sort(key=bit);
oneMiss.sort(key=bit);
twoMiss.sort(key=bit);

return bestMatch + oneMiss + twoMiss;
```

4.3.4 Corner Analysis

Additional filtering is necessary to narrow down the number of ground truth corner identities as potential matches for a Harris corner. The process starts off general then becomes more specific until there is at most one candidate match for any Harris corner. The 3 major steps are: sectioning, analyzing (neighbor and distant neighbor), and removing duplicates.

To perform sectioning, we split up our corners into five sections via six lines. Note that in the rectified view, the top-most and bottom-most Harris corners define the top and bottom horizontal lines. Each corner now has a section that it belongs to.



(a) Harris Corner Section (b) Wrong Section Example (c) Ground Truth Section

Figure 4-9: Corner Sectioning

We section both the Harris corners (Fig. 4-9a) and the ground truth corners (Fig. 4-9c) in order to compare them.

The best-matching candidate (BMC) for a Harris corner is the ground truth corner candidate at the front of the array returned by Alg. 1. Note that any BMC, thus far, is a local-context match as it considers only corners within 2 units of it. Sectioning and distant corner analysis consider and evaluate corners for global matches.

If the BMC falls in a section more than one away from its ground truth section, it is removed from consideration. For instance, if a Harris corner has #151 as its BMC, the Harris corner should belong to section C. If, instead, the Harris corner is located in section A (Fig. 4-9b), the BMC must be wrong and is removed.

Sectioning is a crude first step to eliminate false positive identifications. The next step, corner analysis, uses both the neighbor and the distant neighbor information to support a corner identity. Using the BMCs that remain after sectioning, our algorithm builds a neighbor support system.

Take, for example, the neighborhood around ground truth corner #11 (Fig. 4-10b). This corner has ground truth corner #3 and ground truth corner #4 as its upper and upper-right neighbors, respectively. If a Harris corner (Fig. 4-10a) has #11 as its BMC and its upper and upper-right neighboring Harris corners have #3 and #4, respectively, as their BMCs, it is likely that our algorithm has correctly identified this corner. In this case, this corner has a support score of 2, meaning that its two neighboring Harris corners have BMCs that align with the ground truth corner orientation. As one can imagine, this technique creates corner cliques[4], each of which is a group of Harris corners that support each other's identity. The BMCs that receive a support score of 2 or more proceed to the next step - the distant neighbor analysis.

While the neighbor analysis often finds cliques scattered across the maze, an additional support system of distant neighbors is necessary to justify relative clique locations. This is especially important because cliques are formed using only BMCs

[4]A clique is a subset of vertices in an undirected graph such that every two distinct vertices are adjacent.

(a) Harris Corner Neighborhood (b) Ground Truth Neighborhood

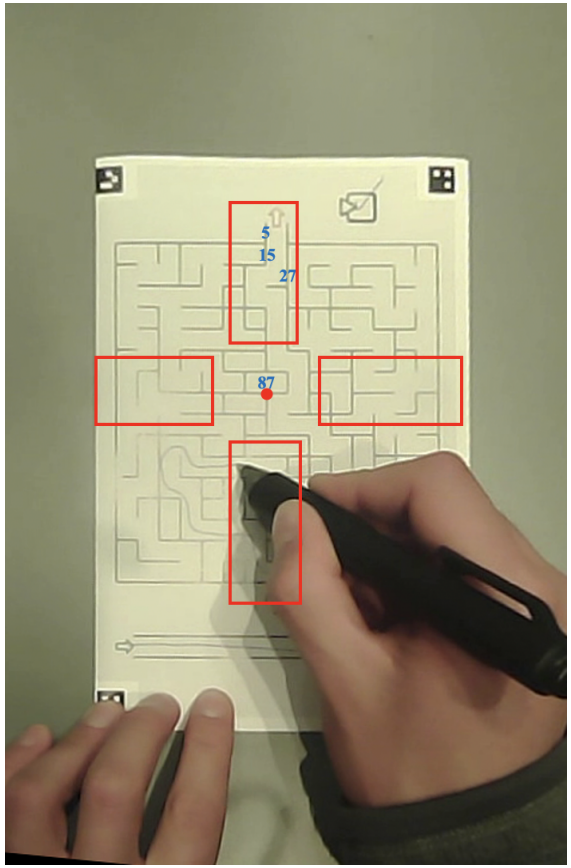
Figure 4-10: Neighbor Analysis

within a small neighborhood. We do not yet know whether these cliques are correctly located relative to each other within the entire maze.

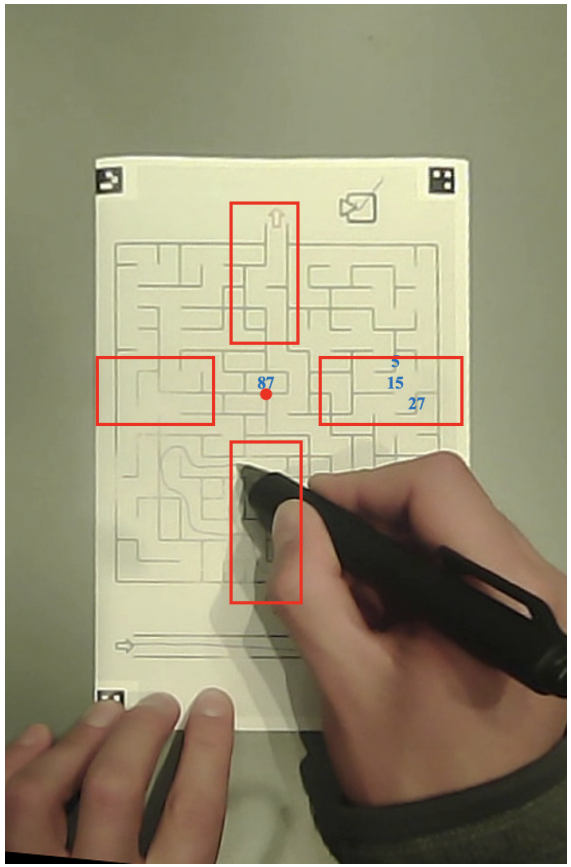
To determine this, we use the distant neighbor information. This focuses on corners far from the corner in consideration. Consider ground truth corner #87 in Fig. 4-11c. It has ground truth corners #5, #15, and #27 far above it. Consider, now, the case where our algorithm has found four Harris corners and has determined that #87, #5, #15, and #27 are their BMCs. Fig. 4-11a shows the situation where Harris corners #5, #15, and #27 are located far above Harris corner #87. In this case, the distant neighbor analysis would award Harris corner #87 a distant score of 3 and award Harris corners #5, #15, and #27 each a distant score of 1.

If, instead, the algorithm has identified the Harris corners in Fig. 4-11b as having BMCs #5, #15, and #27 and they are located to the far right of the Harris corner with #87 as its BMC, Harris corner #87 would now receive a distant score of -3 (it does not match our expectations) while Harris corners #5, #15, and #27 each receive a distant score of -1. Any BMCs with a positive distant score proceed to the next step.

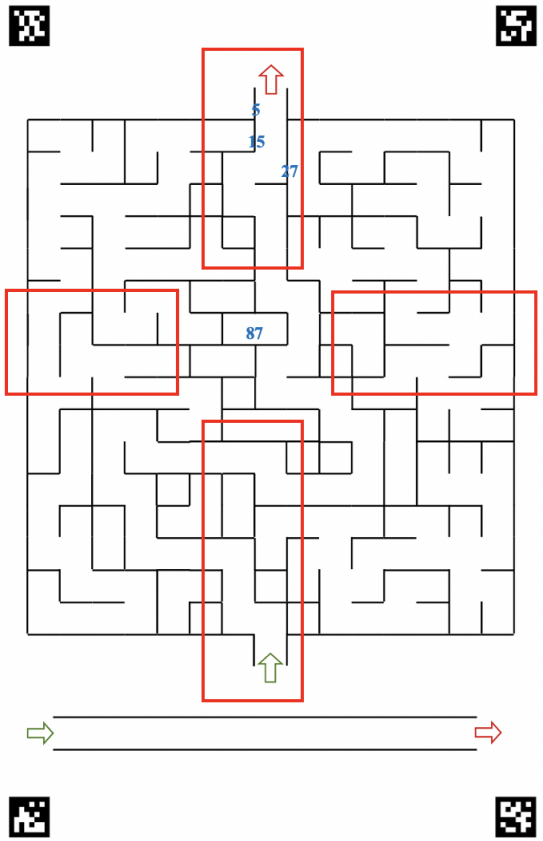
The last step of the corner analysis is to remove duplicate identities. Both support and distant scores are recalculated using the remaining BMCs. For any two corners with the same BMC, the one with a greater sum of support and distant scores is kept. Finally, these BMCs are accepted as the corners' identities.



(a) Correct Distant Neighbor Example



(b) Wrong Distant Neighbor Example



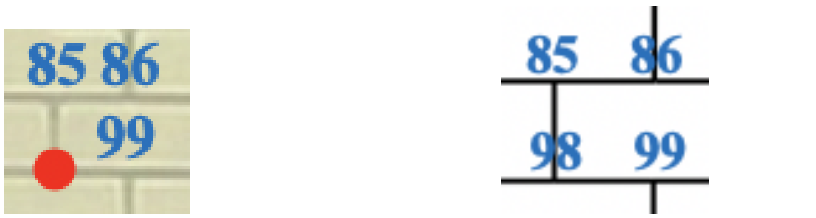
(c) Ground Truth Distant Neighbor

Figure 4-11: Distant Neighbor Analysis

4.3.5 Corner Extension

Given the extensive evidence that is required in order to support the identification of a Harris corner, we can be reasonably confident about the correctness of our Harris corner identities. We use this to further extend our labelling, which allows us to identify additional unlabelled corners.

The algorithm first creates a support system very similar to that of our neighbor analysis. Suppose, as in Fig. 4-12a, Harris corners #85, #86, and #99 have been identified. Suppose, also, that there exists an as yet unidentified Harris corner below Harris corner #85, to the bottom-left of Harris corner #86 and to the left of Harris corner #99. This is where corner #98 should be located given our ground truth neighbor reference (Fig. 4-12b). In this case, this unlabelled corner is hypothesized to be corner #98 with a score of 3, meaning that its three neighboring Harris corners support it being corner #98. A label with a score of 2 or more is accepted. Thus, this technique branches out the labelling even further.



(a) Harris Corner Neighborhood (b) Ground Truth Neighborhood

Figure 4-12: Corner Extension

In each iteration of the corner extension, potentially more Harris corners are identified. We iterate this step until no additional corners may be labelled.

These labelled corners then form the basis for computing a homography to transform the image from the rectified view to the standardized view.

4.4 The Homography Transformation

The homography transformation[5] brings an image from the rectified view to the standardized view. Because different perspectives are linear transformations of one another, we can find the homography to describe the transformation between the rectified view and the standardized view of an image given a set of matching points. The more matching points there are, the more accurate the mapping becomes. Therefore, we iterate between two steps, computing a tentative homography and labelling more corners. When no more corners can be labelled, we have found the final homography.

4.4.1 Homography Computation and Limitation

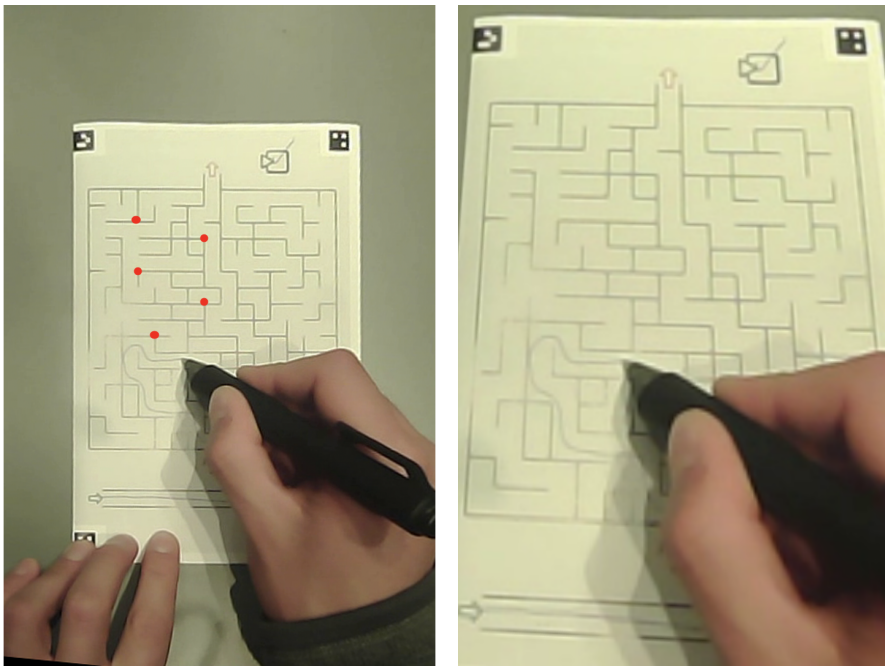
We can define a single homography to transform an image from one view to another if we have four pairs of corresponding points [9]. For more corresponding points, the Least-Median Robust Method is routinely used along with the Levenberg-Marquardt Method to minimize the re-projection error[6] [4, 10, 12]. We chose linear interpolation to ensure that pixels in the standardized view take on integer values.

While the homography describing any transformation is easy to compute, we should keep in mind some of its limitations. Let us first define the points in the original view used to compute the homography as the homography points. Then, the further away a point in the original view lies from these homography points, the less accurate its transformed location is.

This is a problem we sometimes see when computing the homography to map from the rectified view to the standardized view. If our identified corners, i.e., the homography points, span only a small area of the maze (Fig. 4-13a), the standardized view may not match well to the ground truth maze template (Fig. 4-13b). Therefore, we want to pick faraway homography points (Fig. 4-14a) in order to end up with the standardized view that matches well to the ground truth maze template (Fig. 4-14b). Note how much better Fig. 4-14b is compared to Fig. 4-13b.

[5] A homography transformation is the transformation between two images of the same surface.

[6] The re-projection error is a geometric error corresponding to the image distance between a projected point and a measured one.



(a) Nearby Points (b) Standardized View

Figure 4-13: Standardized View from Nearby Homography Points

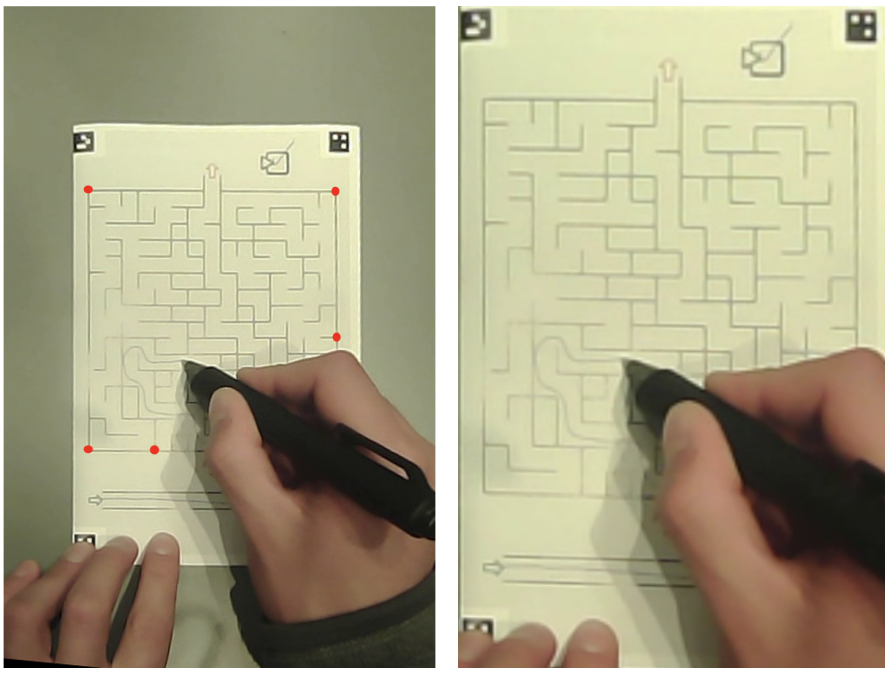
4.4.2 Additional Corner Labelling

To account for the homography limitation, we try to, once more, label unlabelled Harris corners after they have been transformed.

Recall that a transformed pixel is accurate when its location in the original view is close to the homography points. Going off this idea, an unlabelled Harris corner that lies close to the identified corners, i.e., the homography points, is likely to be transformed accurately. If the transformed location of this Harris corner lies less than half the unit to a ground truth corner and no other Harris corners have previously identified to be this ground truth corner, we can label this Harris corner as that ground truth corner.

Additional labelling allows us to expand the area described by the homography points, which leads to a better homography. When no additional corners can be labelled upon transformation, we have found the final, and likely the best, homography.

We conclude the Homography Transformation stage by applying the final homography to the rectified view, effectively getting the standardized view.



(a) Faraway Points (b) Standardized View

Figure 4-14: Standardized View from Faraway Homography Points

4.5 Workflow Summary

We now have every piece of information to transform an image from the world camera view to the standardized view. This allows us to identify where a subject is looking on the maze form without fiducial markers. To summarize, we conclude with the workflow (Fig. 4-15) for our Regular Graphical Pattern Detection Algorithm.

4.6 Outliers

An outlier is defined as a frame that cannot be processed or, having been processed, produces transforms significantly different from those of its previous frame. As previously mentioned, repetition is a fallback technique to compute the vanishing points, the unit, and the homography for frames where these parameters cannot be calculated. When, however, repetition has to be applied to all of these parameters, it means the algorithm is unable to compute anything from the frame alone - everything now depends on the previous frame. The frame in consideration is then an outlier.

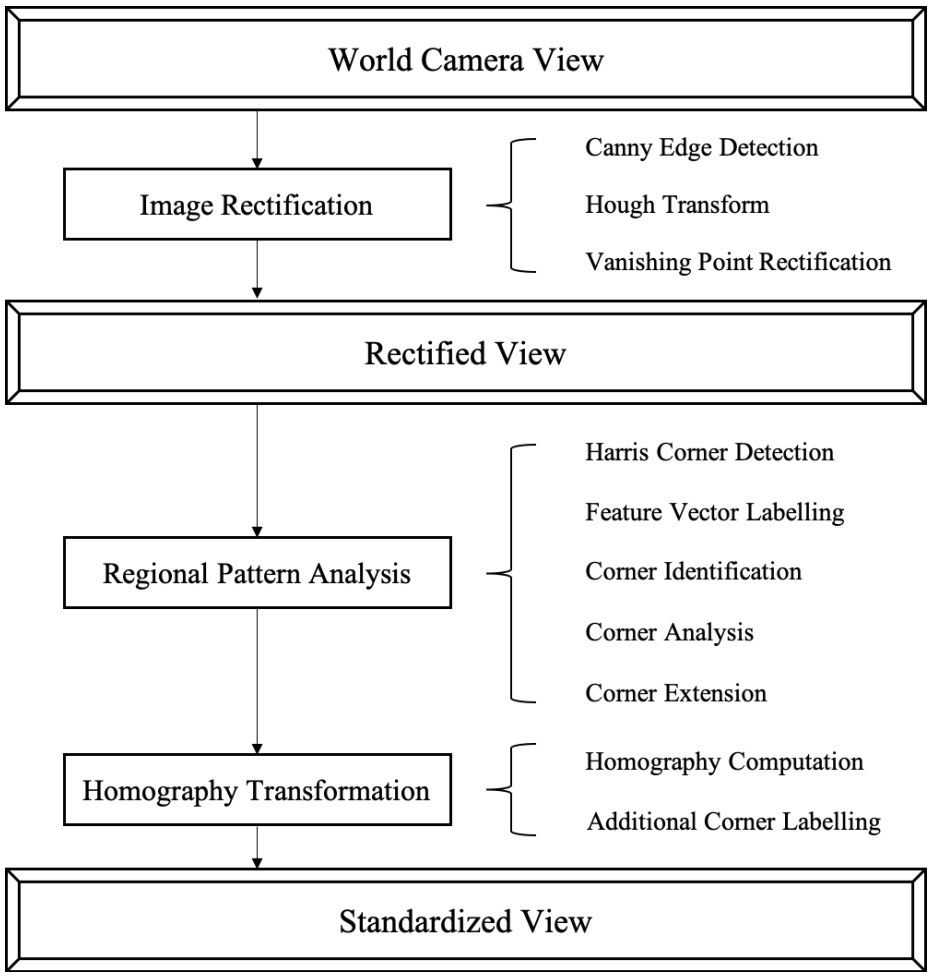


Figure 4-15: Workflow Summary. The Regular Graphical Pattern Detection Algorithm includes 3 views (2 transforms), 3 stages, and 10 sub-stages.

To handle these outliers, we used *forward-reverse processing* and *interpolation* as our repair systems according to Alg. 2.

4.6.1 Forward-Reverse Processing

In an attempt to resolve outliers, we processed the frames in a video in both forward and reverse directions. Because repetition can be applied at various stages of the algorithm, the parameters used to process the current frame may be dependent on its previous frame. An outlier that results during forward processing may in fact not be an outlier when the frames are processed in reverse order. Then, we only needed to interpolate across those that cannot be processed in either direction.

Algorithm 2: HandleOutliers

```
   // RGPD = Regular Graphical Pattern Detection
   // input:  video, output:  outliers
1  define RGPD;
   // input:  outliers, output:  none
2  define Interpolation;

3  ForwardOutliers = RGPD(video, direction='forward');
4  ReverseOutliers = RGPD(video, direction='reverse');

   // only interpolate mutual outliers
5  FinalOutliers = ForwardOutliers.intersect(ReverseOutliers);

6  Interpolation(FinalOutliers);
```

4.6.2 Interpolation

For any frame that could not be handled by either forward or reverse processing, we used the parameter values from the frames before and after to compute its transforms. Furthermore, if consecutive frames required interpolation, we used the nearest frames that do not require interpolation before and after the sequence of outliers to compute transforms for these consecutive outliers. We chose to use linear interpolation and this effectively resolved our remaining outliers.

Chapter 5

Results and Discussion

5.1 Evaluation Metrics

Our primary evaluation metrics are *hit rate*, *sensitivity*, and *goodness of fit*. Hit rate measures the percentage of the frames in a video that can be processed without interpolation. Sensitivity measures the percentage of corners correctly identified out of the actual number of corners visible in a frame. Goodness of fit measures the quality of the standardized view, measured as the median distance between the corners in the standardized view and their ground truth corner locations.

5.1.1 Hit Rate

Along with the hit rate, we also report the temporal distribution of frames missed, i.e., outliers. Consecutive outliers are harder to recover from as they require interpolation between two distant frames and any distortion or shifts not in the direction of the interpolation cannot be recovered.

Consecutive outliers are usually found in a sequence of frames with heavy occlusions, while scattered outliers are usually found in blurred images, typically caused by a sudden shift in the head position.

5.1.2 Sensitivity

Because sensitivity evaluates corner identification, this metric is applicable only for frames where Harris corners have been matched to ground truth corners. Frames whose transforms have been computed via interpolation do not have corner information and are not considered for the sensitivity measure.

Calculating sensitivity is a labor-intensive process because whether a corner has been correctly identified has to be determined manually. As a result, we approximate and report our average sensitivity by evaluating 10% of the frames in each video.

5.1.3 Goodness of Fit

Recall that goodness of fit measures the median distance between corners in the standardized view and the ground truth corner for each frame. The median distance, instead of the average distance, is chosen to account for misidentified Harris corners and is calculated in millimeters to match the width of the pathways when printed (8mm). Then, for each video, we report the average of all the median distances.

5.2 Stress Tests

To evaluate performance for our system, we conducted several stress tests using extreme cases including,

- *rotation test*: rotate test up to 45 degrees, clockwise and counter-clockwise.
- *lean test*: lean forward and backward to capture various world camera angles.
- *shadow test*: cast shadow on the test form with a hand.

The world camera view of an unrotated, unoccluded test form with a participant sitting up straight is the ordinary view (Fig. 5-1). We gradually increased each dimension of the stress individually, followed by a gradual shift back to the ordinary view. The graded increase and decrease mimics real-world (i.e., continuous) body motion and also allows our algorithm to adjust its parameters to better account for the stress.

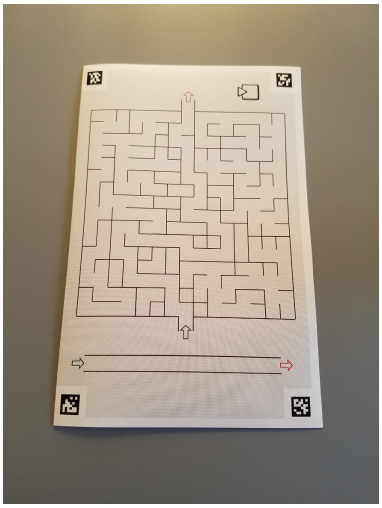
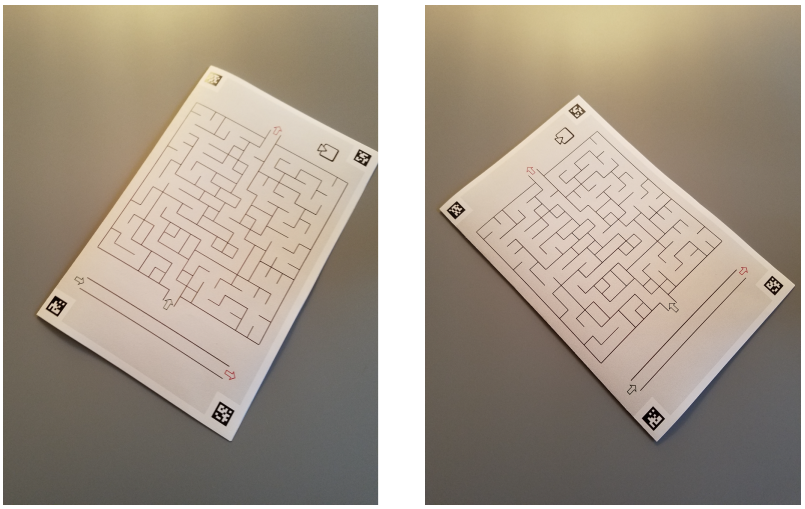


Figure 5-1: Ordinary View

5.2.1 Rotation Test

Image Rectification effectively takes care of the stress caused by rotation. There is no difference in performance between clockwise and counter-clockwise rotations (Fig. 5-2).



(a) Clockwise (b) Counter-clockwise

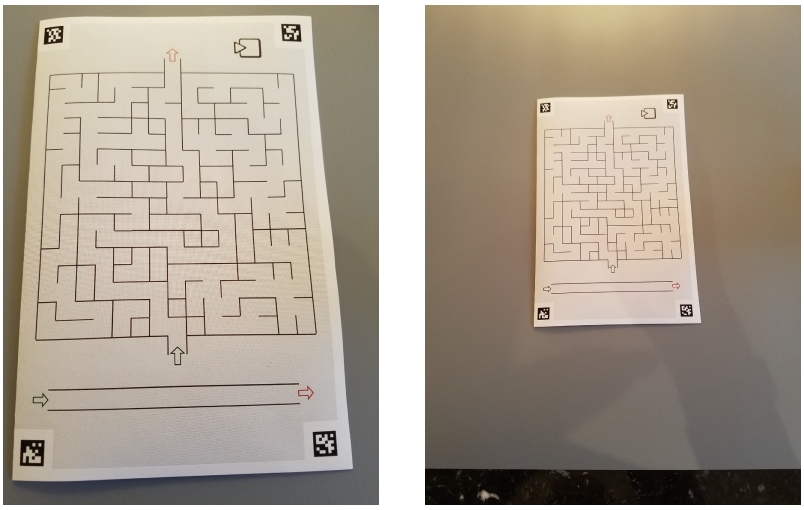
Figure 5-2: Rotation Test

Our algorithm has been tested against rotations up to 45 degrees in both directions because a participant is unlikely to rotate a test form beyond 45 degrees. Nevertheless, the algorithm can be adjusted to work for all degrees if necessary.

In regard to rotations, our algorithm achieved a hit rate of 99.61%, a sensitivity of 94.31%, and an average median distance of 0.85mm. These metrics together indicate good performance with respect to rotations and eliminate rotation as a potential source of error.

5.2.2 Lean Test

Different levels of leaning lead to distinct world camera views (Fig. 5-3). When a participant leans forward, the distance between the world camera and the test form shortens and the view becomes more orthogonal.



(a) Lean Forward (b) Lean Backward

Figure 5-3: Lean Test

Unlike rotations, the effects of leaning forward and backward are not symmetrical. The test form in a lean-forward image is clearer, making feature detection easier. This effect is further exaggerated by bicubic interpolation applied during Image Rectification. Our algorithm can often compute transforms for a lean-forward image but struggles when a participant is leaning far back, away from the test form.

For the lean-forward test, our algorithm achieved a hit rate of 100.00%, a sensitivity of 97.67%, and an average median distance of 0.71mm. For the lean-backward test, our algorithm achieved a hit rate of 20.74%, a sensitivity of 70.58%, and an average median distance of 0.92mm.

5.2.3 Shadow Test

The main difficulty with shadows (Fig. 5-4) originally lies in the step of corner detection. As stated in Chapter 4.3.1, the algorithm first detects corners then filters them by the pre-selected quality level. Given that lines in the maze are black, which leads to black corners, corners under a shadow have significantly lower image intensity gradients. This makes them more difficult to detect than corners that lie outside of the shadow. In response, we lowered the quality level from 10^{-3} to 10^{-4} (our final choice) to accept most corners detected.

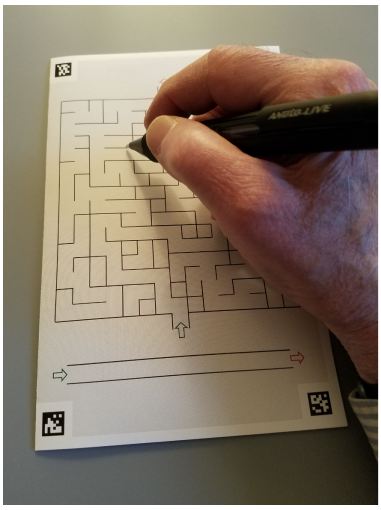


Figure 5-4: Shadow Test

With a lower quality level and the rigorous corner analysis, we were able to effectively detect shadowed corners yet not falsely identifying extraneous corners outside of the shadow. As a result, our algorithm achieved a hit rate of 99.61%, a sensitivity of 93.07%, and an average median distance of 0.95mm.

5.3 Subject Data Results

We processed four videos of the maze test taken by three different individuals. Two of them were recorded at the Lahey Clinic and the other two at our lab. The number of frames ranged from 850 to 1485 frames and each of the frames has a resolution of 1280×720 px.

The evaluation results for each video are provided in Table 5.1. On average, our algorithm achieved a hit rate of 89.75%, a sensitivity of 90.00%, and an average median distance of 1.27mm.

Data Source	Lahey 1	Lahey 2	Lab 1	Lab 2
Hit Rate (%)	85.93	82.96	90.22	99.89
Sensitivity (%)	84.61	84.85	92.78	97.76
Goodness of Fit (mm)	1.69	1.64	0.92	0.81

Table 5.1: Subject Data Evaluation Results

The temporal distributions of outliers for each of the video are provided in Fig. 5-5. The result is binary (i.e., an outlier or not) for each frame.

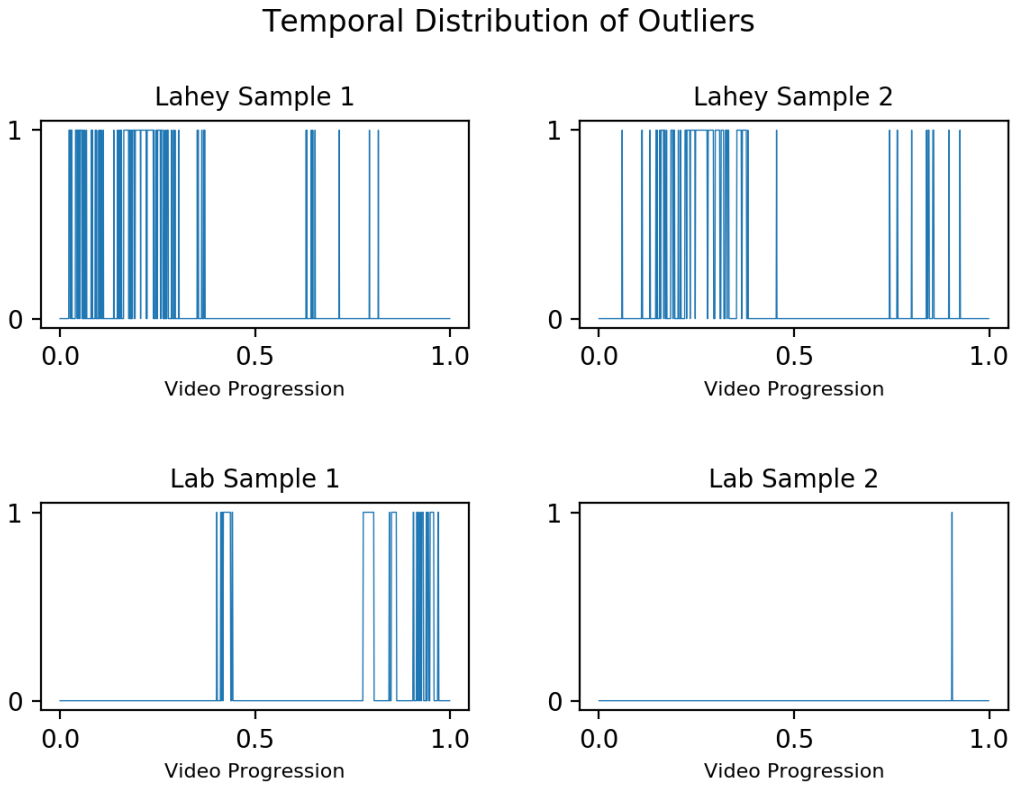


Figure 5-5: Temporal Distribution of Outliers

The frequency distributions of consecutive outliers for each of the video are also provided in Fig. 5-6. This allows us to better understand the way outliers happen.

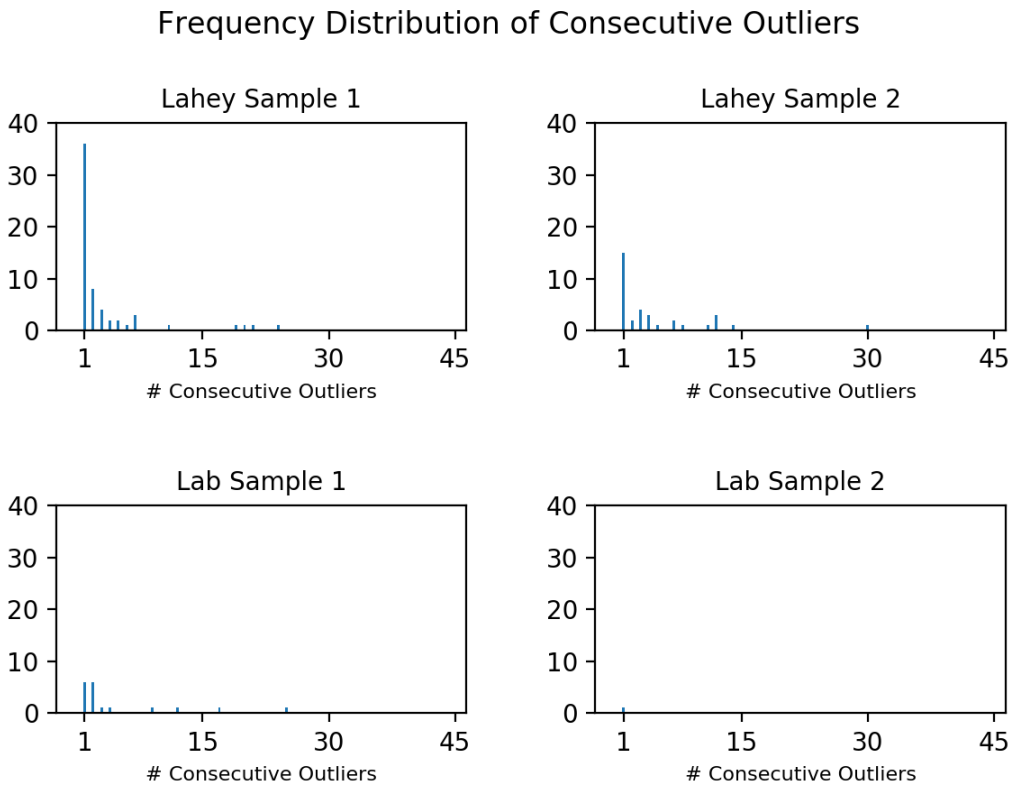


Figure 5-6: Frequency Distribution of Consecutive Outliers

5.4 Discussion

The results in Table 5.1 show that our algorithm achieved at least 80% hit rate across all of our subject videos. It is important to note that the performance with respect to the videos taken at the Lahey Clinic is overall lower due to some administrative error. Specifically, the world camera on the eye tracker was not adjusted well, causing some of the test form to be cut off for a limited number of frames. As a result, our algorithm was unable to detect features necessary to compute the transforms.

The sensitivity achieved suggests that most of the identified corners actually had correct labels. At the same time, there were far more than four corners identified for any frame, allowing us to use many corners to compute the homography. We can, therefore, be less concerned about the limitation of homography computation - the vast number of homography points likely span most of the maze.

Recall that our research aims to achieve an accuracy that is within half a degree of the visual angle. This corresponds to an error bound that is approximately half the width of a path in the maze form, which, as noted above, we call a unit. Therefore, the goodness of fit achieved indicates that we have successfully accomplished our goal. Our algorithm detected gaze position to be within 20% of the unit from its true location.

The temporal (Fig. 5-5) and frequency (Fig. 5-6) distributions further support the use of interpolation in handling outliers, as outliers are typically scattered across a video. There was only one occasion (in Lahey Sample 2) where there were thirty consecutive frames missed, which ultimately constituted only one second. All others had outliers that lasted for less than one second of the video. Thus, we can be reasonably confident about the robustness of our algorithm.

Chapter 6

Conclusions and Future Work

6.1 Contributions

This thesis presents a new object detection method and effectively applies it to a patterned form. It is a multi-stage image processing algorithm that aims to transform an occluded, patterned image in an arbitrary view, back to its standardized view. We used vanishing points to correct an image, allowing us to undo perspective skew. Then, we detected corners and created patterns out of their neighboring corners, making each of them uniquely identifiable. These identified corners were then used to compute a homography, re-projecting our image back to its standardized view.

We tested our algorithm on four subject videos, with frame counts ranging from 850 to 1485, and one stress test video, consisting of rotation, lean, and shadow. For all of these videos, our algorithm achieved high accuracy in identifying the test form in an image. Then, for frames that cannot be processed, analyses in Chapter 5.4 further support the use of interpolation to take care of them, making our algorithm an overall robust technique to identify patterned forms.

6.2 Achievements and Shortcomings

The outputs of these videos demonstrate success in our algorithm, where success is defined by three evaluation metrics: hit rate, sensitivity, and goodness of fit.

While our algorithm performs well on most frames of our videos, in the case of heavy occlusions, consecutive frames are missed. Interpolation allows us to recover results between two good frames, but fails to capture any sudden shifts in position. That is, our algorithm is unable to account for the case where, for instance, the test paper was shifted to the right and back during the period of heavy occlusions. At the same time, our algorithm depends on features built on local neighborhoods. It is, therefore, unlikely to perform well in the presence of multiple scattered occlusions, an uncommon situation given we have mostly continuous objects (e.g., arms and pencils). Nevertheless, these remain challenges our algorithm has not yet overcome.

6.3 Future Work

For future work, we plan to improve our methods in three aspects, each of which addresses slightly different parts of our algorithm.

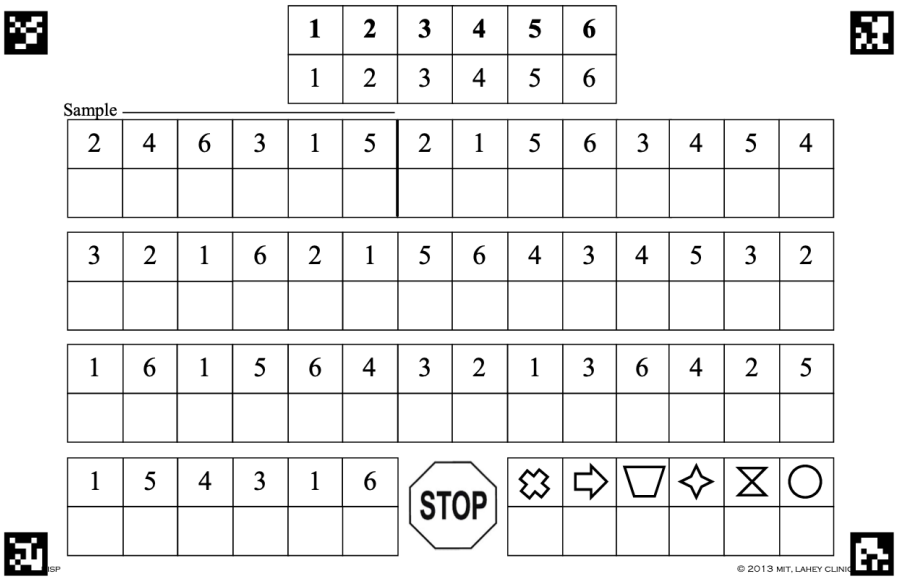
We have noticed that multiple missed frames come from failures at the Image Rectification stage. Our algorithm sometimes struggles to detect lines suitable for determining the vanishing points. Therefore, we suspect it would be useful to create our own line detection algorithm, perhaps an extension of the Hough transform.

Aside from consecutive missed frames, we have noticed that scattered missed frames are usually results of blurred images, which are commonly due to a sudden shift of the user's head. Using filters like the box filter might sharpen the image, thereby allowing our algorithm to process it successfully. However, we have yet to find and incorporate a suitable filter.

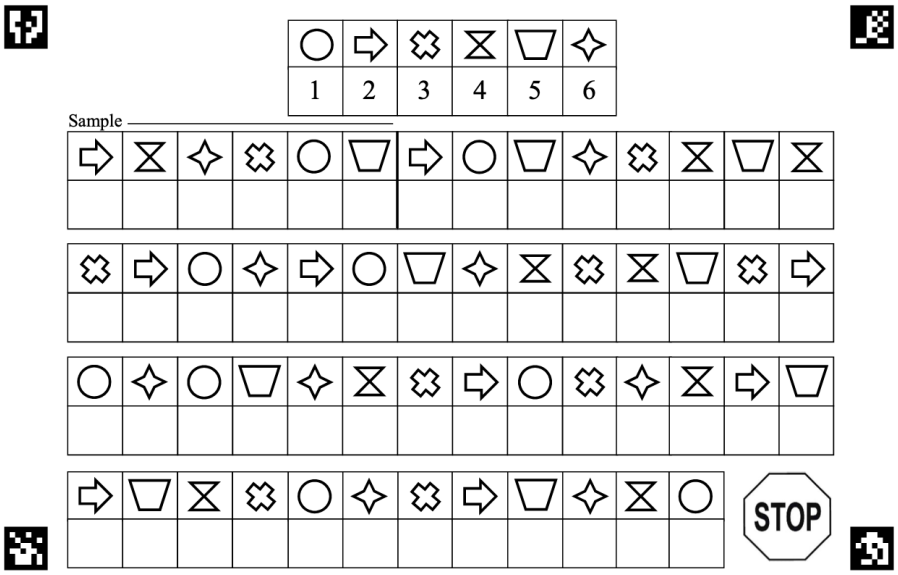
We have seen and shown here the effectiveness of identifying corners using local neighborhoods. Therefore, we believe we can further extend our approach to various types of neighborhoods. Instead of depending solely on local neighborhoods to identify each corner, our algorithm can be more robust against occlusions if we create feature vectors that label and identify corners using multiple different references. We are confident that such advancement will push the capabilities of our algorithm and object detection even further.

Appendix A

Figures



(a) The LCL Test



(b) The HCL Test

Figure A-1: The Symbol Digit Tests. The symbol-digit test is a grid-like mapping test. It requires a subject to write the corresponding number for each symbol in the test.

Bibliography

[1] Claire Lancaster Chris Hinds Ivan Koychev Amy Chinner, Jasmine Blane. Digital technologies for the assessment of cognition: a clinical review. *Evidence-Based Mental Health*, 21(2):67+, May 2018.

[2] Alzheimer's Association. 2019 alzheimer's disease facts and figures report, 2019.

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:679+, November 1986.

[4] Peter J. Rousseeuw Annick Leroy Desire L. Massart, Leonard Kaufman. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, 187:171+, January 1986.

[5] Martial Hebert Edward Hsiao. Occlusion reasoning for object detection under arbitrary viewpoint. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146+, 2012.

[6] Hae-Jeong Park Hyunjoo Song, Do-Joon Yi. Validation of a mobile game-based assessment of cognitive control among children and adolescents. *PLOS One*, March 2020.

[7] S.Y. Chen-Qiu Guan Ke Zhang Jianhua Zhang, Y.F. Li. Partial occlusion detection of object boundary. *IEEE Instrumentation and Measurement Technology Conference*, pages 30+, July 2009.

[8] Zhishuai Zhang-Jun Zhu Lingxi Xie Alan Yuille Jianyu Wang, Cihang Xie. Detecting semantic parts on partially occluded objects. *Computer Vision and Pattern Recognition*, July 2017.

[9] David Kriegman. Homography estimation. UCSD Computer Vision I CSE 252A Lecture Notes, 2007.

[10] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164+, July 1944.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91+, November 2004.

[12] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431+, June 1963.

[13] Hans P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Tech report, Carnegie-Mellon University, Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, September 1980.

[14] Erum Naqvi. Alzheimer's disease statistics. Alzheimer's News Today, June 2017.

[15] National Institute on Aging. What causes alzheimer's disease? December 2019.

[16] Peter E. Hart Richard O. Duda. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), January 1972.

[17] Qingshan Liu Shengye Yan. Inferring occluded features for fast object detection. *Science Direct*, 110:188+, May 2015.

[18] Xiao Bian-Zhen Lei Stan Z. Li Shifeng Zhang, Longyin Wen. Occlusion-aware r-cnn: Detecting pedestrians in a crowd. *EECV*, pages 657+, October 2018.

[19] Moises Betancort Alejandra Machado Maria Lindau Stina Bjorngrim, Wobbie V. Hurk. Comparing traditional and digitized cognitive tests used in standard clinical evaluation – a study of the digital application minnemera. *Frontiers in Psychology*, 10:2327+, October 2019.