3D Scene Angles using UL Decomposition of Planar Homography

Vikas Paliwal
vpaliwal@stanford.edu

Abstract

Proctoring during online exams often requires students to be under surveillance from a side pose and there is a strong need to estimate the side camera's relative position with respect to student's computer screen. This work uses edge and line detectors to extract the computer screen's boundaries and estimates homography with respect to rectangular shape with corresponding aspect ratio as in a normal view. A novel Upper-Lower Decomposition of Homography (ULDH) algorithm is proposed that calculates the polar and azimuthal angles with less than 5^o mean errors and can help distinguish bad camera placements from good ones with good precision. A purpose-built dataset is created and validated for this purpose and the software for key parts of image processing pipeline is made available for remote proctoring purposes.

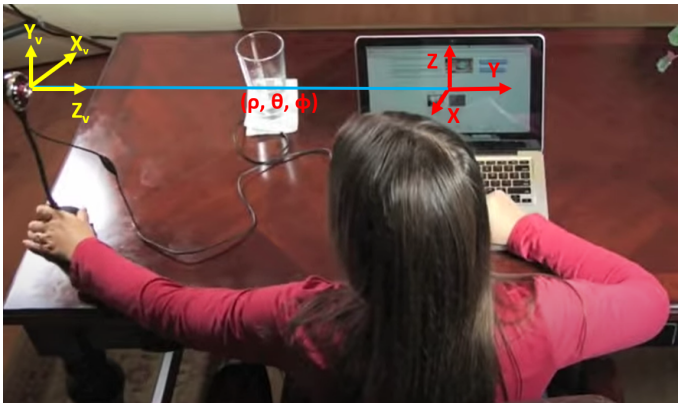
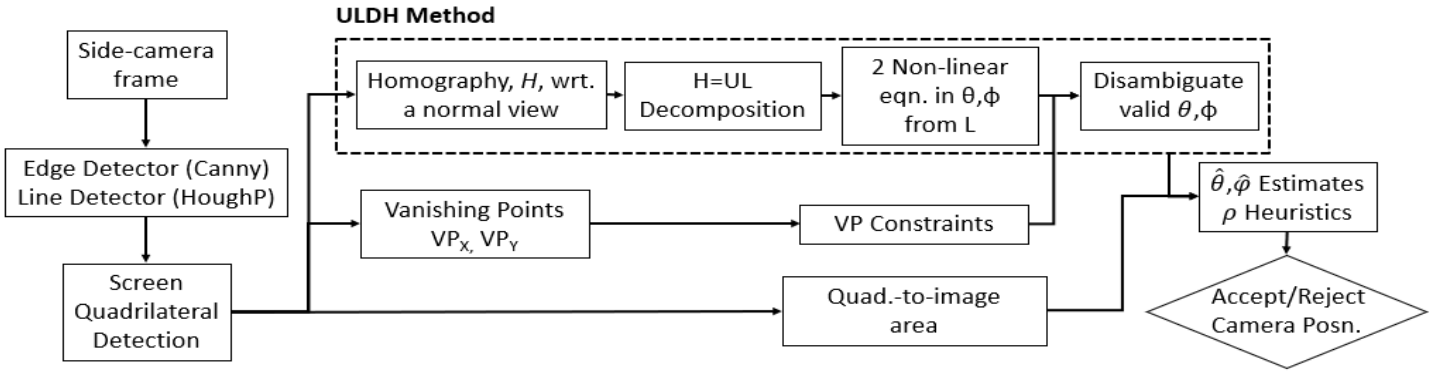


Figure 1: Sample camera setup and scene geometry, *source*

1. Introduction

COVID-19 related social distancing and general trend towards remote learning has brought on a challenge on maintaining integrity during online exams. As such, there is a greater need to ensure students do not refer to Internet during the exam or receive other person's assistance. Thus many experts in education fields strongly recommend using a second side camera that focuses on the student and his/her device screen[1, 2]. This ensures that a human proctor can look at candidate's work area including the computer screen at any time. However the device screen is not visible properly unless the side camera is positioned suitably. Due to cost of human proctors, many testing services only check the camera settings at start of exam and only in certain cases a human proctor surveils the scene for the entire test duration. This is clearly sub-optimal and recent advances in computer vision can definitely be used to build solutions that can automate the proctoring services. This work focuses on one such aspect of online proctoring i.e. checking if screen is not too oblique or too far from side camera so that the side view is good for checking candidate's activity on computer screen. Mathematically, this is equivalent to decision-making based on polar angle, θ, azimuthal angle, ϕ, and distance, ρ, in the polar coordinate system with YZ plane on screen surface and X axis normal to the surface as shown in Figure 1. Based on estimation of good viewing angles, we wish to define thresholds for good polar/azimuthal angles as, $20^\circ < |\theta| < 70^\circ$ and $55^\circ < \phi < 125^\circ$. In doing so, the expectation is that an automated proctor developed using such an approach can detect when a device screen is not detected or not properly oriented in view of side camera. Further, this could also assist test-takers to initially set their side camera for the exam. In this work, we propose that the rectangular shapes of computer screens and their typical known aspect ratios can be exploited to decipher the scene geometry. The side view image is first run through a Canny Edge Detector [3] to get all the edges in the scene, followed by doing probabilistic Hough line transform [8] to extract the linear edges. A method to detect the projectively transformed rectangle as a quadrilateral is implemented that identifies the screen's visible boundaries. A homography is then found and decomposed using proposed ULDH algorithm and values of scene angles are estimated from a pair of non-linear equations. Further, the camera-screen distance/depth is tested against quadrilateral to image area heuristics. The scene angle estimates from ULDH, together with distance heuristics, is then used to accept or reject a specific camera position.



.

Figure 2: Pipeline for camera geometry estimation

2. Related Work

The scene geometry estimation problem can leverage some existing widely used approaches like Canny Edge Detector [3] and probabilistic Hough line transform [8] but to the best of our knowledge there is currently no usable approach to derive scene geometry angles from planar homography without knowing the camera's intrinsic calibrartion matrix. Interest in this domain started from the early work by Faugeras and Lustman [5], where they show a method to perform singular value decomposition (SVD) on the homography matrix and thereby derive the translation, rotation and normal vector of the scene plane. Further, more recent work in this domain was shown by Malis and Vargas [6] where a new analytical method to decompose the homography matrix is presented and the method ensures ambiguity among multiple candidate solutions can always be resolved. This method is understood as a widely used method and is the one currently used by OpenCV's *decomposeHomographyMat()* [10]. It should however be noted that these and multiple other approaches presume that camera calibration, K, is known or can be calculated. For many cases such as ours where a single view is available from a remote camera that is not physically accessible, such approaches are unrealistic.

If we turn our attention to other approaches with uncalibrated cameras such as Hartley [7] where the focus is on deriving full scene geometry viz. two cameras' calibrations as well as rotation and translation, the author offers general form solutions in relative terms. This work recognizes identifying both camera calibration and extrinsic parameters is too much to hope for but we see that the proctoring scene qualification problem is far simpler in primarily requiring only finding the two rotation angles with respect to screen coordinate system as shown in Figure 1. A work somewhat close to what our ULDH algorithm attempts to do is shown in [8] where authors categorize the scene geometry in 12 bins and classify images pulled from Internet to a suitable bin. While such binning based methods could have offered partial solution to our specific case, but a more precise algorithm to estimate scene angles from a single image, while knowing 4 point correspondences w.r.t. to a normal view and hence the homography, is not available to the best of our knowledge. Regarding screen-camera distance, there has been lots of recent work on monocular depth estimation but we felt for this problem this would have been an overkill and decided to work with a simple distance heuristic based on image magnification and occupied area in the image.

3. Proposed Approach

Figure 2 shows the overall approach for determining an understanding of the scene geometry. The parameters for the flow may be tuned for specific image size, so initially the image is resized to a target size for which the processing pipeline is tuned. This implementation primarily works with images of sizes in 600 to 1000 pixels in terms of length and width. The first step involves feeding an image frame to Canny edge detector [3] available through OpenCV. This should yield all the detected edges in the image. This is followed by OpenCV's probabilistic Hough Line transform [4] to detect linear edges in the scene. Next step is to determine the quadrilateral from these lines which is formed from the most likely candidate for device screen edges. Since it is known that a rectangular screen's edges will have two pairs of parallel edges, so two vanishing points can be established from the quadrilateral. These vanishing provide us constraints on the polar and aziumthal angles based on their position with respect to quadrilateral's centroid. Further, based on known or estimated aspect ratio of monitor screen, a corresponding rectangle pivoted to longest edge of quadrilateral is overlaid to the quadrilateral to determine the planar homography between a side camera view a normal view from certain distance, which we do not calculate ex-

plicitly. Subsequently, based on proposed ULDH method, an UL (upper-lower triangular) decomposition is done on the homography to split the offset portion of camera matrix and product of diagonal portion of camera matrix and 3D viewing transformation matrix. The lower triangular matrix from this decomposition yields two non-linear equations that can be solved to yield two candidate solution in the valid range of polar and azimuthal angles. The vanishing point criteria can be used to pick the valid solution. Then we use our predefined thresholds, $20° < |\theta| < 70°$ and $55° < \phi < 125°$ to accept or reject a camera placement based on angles. Additionally, the ratio of the area of quadrilateral to the overall image area is indicative of the relative distance of screen from camera and a heuristic threshold is used to distinguish cameras that are placed too far. Overall the three methods - ULDH, vanishing points and area ratio - provide insights into scene geometry and are used to accept or reject a specific side camera placement with respect to device screen.

3.1. Edge/Line Detection

For the initial overall edge detection in the image, the Canny edge detector is used with default configuration. Using this, all the relevant edges are extracted. From this point, the task is to determine all the linear edges in the scene. Based on experimentation, it was determined that it was better to work with probabilistic Hough Line transform due to eventual need for actual end points of line segments. It is noteworthy to mention that parameters of probabilistic Hough transform need to be chosen carefully. Due to possible noise or camera lens distortions, often a known single line would come up as set of jagged segments from Canny detector. The Hough transform accumulator threshold setting for *maxLineGap* is kept at 20 pixels to stitch such split edges while relative angular *tolerance* is kept low at 0.5 radians to pick multiple lines. The *minLineLength* is kept at 80 pixels. All these settings are chosen based on expected line lengths and its variability for image shapes in 800×800 pixel range. In certain cases, where subsequent steps fail, the Hough transforms can be re-run with progressively decreasing *minLineLength*, while keeping in mind that smaller line thresholds likely result in more line candidates in the scene and lead to slower performance.

3.2. Quadrilateral Detection

As shown in Algorithm 1, the lines returned from probabilistic Hough Line Transform are used to make a suitable quadrilateral. The quadrilateral is built in a step-by-step manner under the assumption that all 4 edges are clearly visible. In case of extra clutter or color-blend issues, a sentinel pattern can be displayed by the web conferencing service provider viz. Zoom on the screen so that side camera immediately identifies the screen. It is expected that screen area will be surrounded by monitor's border so the approach eliminates any enclosing quadrilateral and returns the non-enclosing quadrilateral with largest area in the image.

Algorithm 1 Calculate screen's image quadrilateral

Input: $lines$ with endpoints from HoughLineTransformP, slope/distance tolerances s, d
Output: $vertices$ of likely quadrilateral
  while all $lines$ pairs not visited **do**
    make $linepairs$ with $line_i$ and $line_j$ with slope $> s$, endpoint separation $< d$
  end while
  while all $linepairs$ and $lines$ not visited **do**
    make $linetriplets$ with $lines_k$ with slope $> s$, endpoint separation $< d$ from one of $linepairs_{ij}$ endpoints
  end while
  while all $linetriplets$ and $lines$ not visited **do**
    make $linequads$ with $lines_l$ with slope $> s$, endpoint separation $< d$ from one of $linetriplet_{ijk}$ endpoints
  end while
  Return non-enclosed quadrilateral with largest area

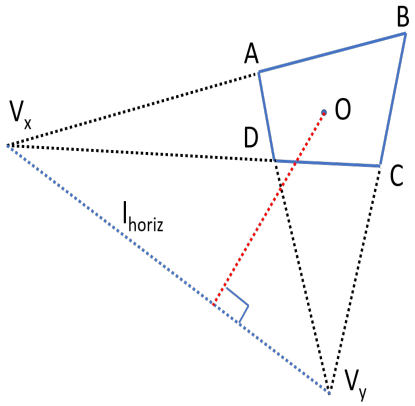


Figure 3: Determining the horizon line for device screen plane in image

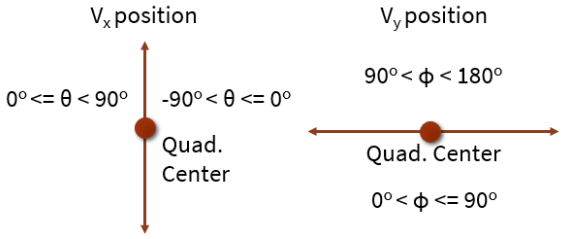


Figure 4: Polar and azimuthal angle relation to vanishing points

3.3. Single View Metrology

From the projective geometry it is understood that the device monitor surface is a plane so if both of the length and width sides are captured, then we get two vanishing points from scene parallel lines in two perpendicular directions. Such analysis can be extended to as shown in Figure 12 where after determining the vertices A, B, C, D of the quadrilateral, the projections of the two parallel sides are extended to intersect in the vanishing points V_x and V_y corresponding to parallel edges on length and width sides. The vanishing point that is at a greater distance from quadrilateral center is less projectively distorted and if we check if its x or y coordinate absolute value is greater, then it is the vanishing point along that axis and the other vanishing point must be along the other axis. From these two points, the horizon line that contains all possible directions in device screen plane, l_{horiz}, is determined as $l_{horiz} = V_x \times V_y$ in the homogeneous coordinates. It can be understood that distance and orientation of this horizon line from center of quadrilateral is related to camera position. If the camera is on left to the screen i.e. $-90° < \theta < 0°$, the vanishing point along length, V_x will appear on the right and vice versa for a camera on the left. Similarly if camera is above the screen center i.e. $0° < \phi < 90°$, the vanishing point along the width of the screen will appear below the quadrilateral center in the image and vice versa. These constraints are shown in Figure 5 and will be later used to resolve ambiguities in the estimated angles. It must be mentioned though that we later end up with only two candidate solutions of θ, ϕ so only one of θ or ϕ constraints can help resolve the ambiguity as we will see later and we use the constraint generated by the vanishing point closer to quadrilateral center, i.e. more mid-range values in solution space subdomains shown in Figure 5 and not borderline values around $\theta = 0°$ or $\phi = 90°$.

3.4. The ULDH Method

The top portion of Figure 2 shows the key elements of Upper-Lower Decomposition of Homography (ULDH) approach. Now we describe the details of each of the step involved in this proposed method.

3.4.1 Homography Estimation

For calculating the scene angles, we propose to calculate the planar homography between the camera view and a view normal to the device screen plane i.e. along the x axis in Figure 1. We employ the four vertices of the rectangular screen in the normal view and corresponding vertices of the quadrilateral in the camera image. One important consideration here is that for faithful representation of metric rectangular shape, the aspect ratio should be exact or close. We propose that since most computer monitors employ standard aspect ratios like 16:9 or 16:10, we can use this info to draw

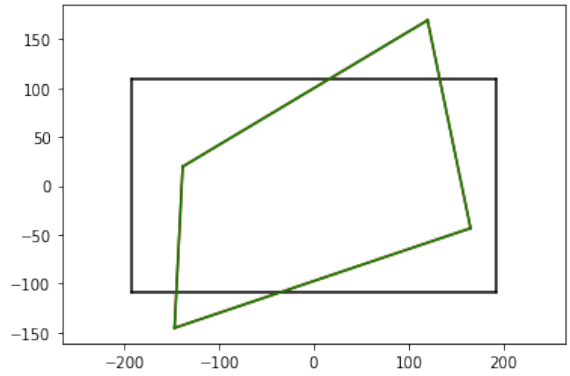


Figure 5: A sample rectangle overlaid on quadrilateral to determine homography

the normal view rectangle. Alternatively, the video conference tool provider for the online proctoring service that controls both candidate's main screen and side camera, has the knowledge of the main screen's aspect ratio and can use that for side view homography. Although the scale of rectangle doesn't matter, for our implementation, we just draw a rectangle pivoted to largest edge of quadrilateral and other two adjacent sides determined by the aspect ratio. This point correspondence is fed to OpenCV's *findHomography()* [10] method where the source points are from the normal rectangular view and destination are quadrilateral image vertices and a homography H is determined.

3.4.2 UL Decomposition

It should be noted that homography found in previous step is projective homography (i.e. including the influence of camera intrinsics) so we need to use a suitable camera matrix. We choose a simpler yet realistic 3-parameter camera intrinsic model with paramters f, c_x, c_y for the focal length and x, y offsets of the principal point in the image plane, i.e. $K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$. Now it has been shown in [9], that the 3D viewing transformation under the assumption of z axis of screen coordinate system defining the upwards direction is defined by, $P_v = \begin{bmatrix} -sin\theta & cos\theta & 0 & 0 \\ -cos\phi \cdot cos\theta & -cos\phi \cdot sin\theta & sin\phi & 0 \\ -sin\phi \cdot cos\theta & -sin\phi \cdot sin\theta & -cos\phi & \rho \end{bmatrix}$.Then we can find the homgraphy between the normal view and the side camera view by recognizing that by our choice of screen coordinate system in YZ plane, the X value of 3D point P in screen plane is zero i.e. $P = [0, Y, Z, 1]^T$. So the relation between image point $p = [x, y, 1]^T$ and screen plane 2-D point $\tilde{P} = [Y, Z, 1]^T$ is defined by a homography H between the two views. We can derive this homography H explained

through the 3D viewing and camera projection transformations as shown in eqn. 1. It can be verified from eq. 1 that for normal view, $\theta = 0°, \phi = 90°$, the euclidean homography (right matrix in matrix multiplication) becomes a $diag\{1, 1, \rho\}$ matrix, or a simple metric scaling as we would expect to be the case. Eqn. 1 also shows that the homography matrix is composed as a product of an upper triangular and a lower-triangular matrix of forms shown in last step of the equation. We can use this fact to decompose our estimated homography from previous step to decompose H as a product of an upper and a lower triangular matrix. Since Python and Numpy do not offer a recipe to readily perform a UL decomposition as desired, a UL decomposition for a 3×3 matrix is implemented as a product of two transformations T_1 and T_2 formed using Gaussian elimination/row-reduction techniques as shown in eqn. 2.

$$
\begin{aligned}
H\tilde{P} = p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= KP_vP = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \\
&\begin{bmatrix} -sin\theta & cos\theta & 0 & 0 \\ -cos\phi \cdot cos\theta & -cos\phi \cdot sin\theta & sin\phi & 0 \\ -sin\phi \cdot cos\theta & -sin\phi \cdot sin\theta & -cos\phi & \rho \end{bmatrix} \begin{bmatrix} 0 \\ Y \\ Z \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & 0 \\ -cos\phi \cdot sin\theta & sin\phi & 0 \\ -sin\phi \cdot sin\theta & -cos\phi & \rho \end{bmatrix} \begin{bmatrix} Y \\ Z \\ 1 \end{bmatrix} \\
= \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix} &\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & 0 \\ -cos\phi sin\theta & sin\phi & 0 \\ -sin\phi sin\theta & -cos\phi & \rho \end{bmatrix} \begin{bmatrix} Y \\ Z \\ 1 \end{bmatrix} \\
= \rho \begin{bmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{bmatrix} &\begin{bmatrix} \frac{f}{\rho}cos\theta & 0 & 0 \\ -\frac{f}{\rho}cos\phi \cdot sin\theta & \frac{f}{\rho}sin\phi & 0 \\ -\frac{1}{\rho}sin\phi \cdot sin\theta & -\frac{1}{\rho}cos\phi & 1 \end{bmatrix} \begin{bmatrix} Y \\ Z \\ 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

$$
H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}, T_1 = \begin{bmatrix} 1 & 0 & -h_{13} \\ 0 & 1 & -h_{23} \\ 0 & 0 & 1 \end{bmatrix}
$$

$$
\implies H^{'} = T_1 \cdot H = \begin{bmatrix} h_{11}^{'} & h_{12}^{'} & 0 \\ h_{21}^{'} & h_{22}^{'} & 0 \\ h_{31}^{'} & h_{32}^{'} & 1 \end{bmatrix}
$$

$$
T_2 = \begin{bmatrix} 1 & -\frac{h_{12}^{'}}{h_{22}^{'}} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \implies H^{''} = T_2 \cdot H^{'} = \begin{bmatrix} h_{11}^{''} & 0 & 0 \\ h_{21}^{''} & h_{22}^{''} & 0 \\ h_{31}^{''} & h_{32}^{''} & 1 \end{bmatrix}
$$

$$
U = T_1^{-1} \cdot T_2^{-1}, L = H^{''} \tag{2}
$$

It is worth mentioning that since the UL decomposition of eqn. 2 is not unique and any diagonal matrix D can be inserted in between $H = UL = UDD^{-1}L$ so the subsequent analysis we do on first two rows of lower triangular matrix is dependent on our initial assumption of square pixels ($f_x = f_y$).

3.4.3 Non-Linear Equation Setup

Noticing the elements of the lower triangular matrix L, l_{ij} derived from eqn. 2 and its expressions in eqn. 1, we observe that,

$$
L = \begin{bmatrix} \frac{f}{\rho}cos\theta & 0 & 0 \\ -\frac{f}{\rho}cos\phi \cdot sin\theta & \frac{f}{\rho}sin\phi & 0 \\ -\frac{1}{\rho}sin\phi \cdot sin\theta & -\frac{1}{\rho}cos\phi & 1 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \tag{3}
$$

Since first two rows are scaled up by the focal length, the third row entries of L are less useful for further analysis so we focus for the three non-zero elements in the first two rows. Without knowing the f/ρ value, we can divide the entries and use ratios to setup two non-linear equations as,

$$
l_{22} \cdot cos\theta - l_{11} \cdot sin\phi = 0 \tag{4}
$$

$$
l_{22} \cdot cos\phi \cdot sin\theta + l_{21} \cdot sin\phi = 0 \tag{5}
$$

These two non-linear equations in two unknowns θ and ϕ can be solved using non-linear solvers such as Python's *fsolve()* routine with an initial estimate at mid-point of our valid range, i.e. $\{(\theta, \phi) = (0°, 90°)\}$.

3.4.4 Solution Disambiguation

The solution from non-linear solvers can often go beyond the valid range due to repetitive patterns of trigonometric functions but they can be brought in target range through modulus operation. However, even within the valid solution range $\{-90° < \theta < 90°, 0° < \phi < 180°\}$, it can be seen that two solutions can simultaneously satisfy eqn. 4 and 5 and we can see that if (θ, ϕ) is a solution then $(-\theta, 180° - \phi)$ is also a solution. It is here that the vanishing point constraints established in section 3.3 can be put to use and the unique valid solution for (θ, ϕ) is obtained and the other solution is easily discarded. This disambiguation can also be done by applying the homography H to our rectangular shape point and the correct solution should yield the original quadrilateral while the invalid solution will not.

3.5. Distance Heuristics

Since a precise distance estimation is not necessary for this application, we propose to use an approximation approach that uses the ratio of quadrilateral area to overall image ratio i.e. metric QA defined as, $QA = \frac{ScreenImageArea}{TotalImageArea}$ as a heuristic for the distance. It can be understood that, in normal view, the area decreases as inverse of the square of distance so we set a threshold below which the device screen is declared to be too far from side camera. For our tests, we

used a threshold of 7.5% based on subjective survey of proctors on the size below which they will declare a camera as too far.

4. Experiment/Results

In order to validate the efficacy of the proposed approach, tests are performed against a dataset of images created for this purpose. We discuss the results of such testing and processed image outputs generated by the code written based on suggested approach.

4.1. Experimental Methodology/Dataset

As shown in Appendix B, we use ruler, compass and angle gauge for angle measurements. On the flat surface under the monitor a paper is annotated with guiding rays for specific θ angles. On the screen monitor, a crosshair position is shown on the center of display area. Using the crosshair, guiding ray on table surface and a compass underneath rulers of specific lengths $\rho = 3', 4.5^{'}, 7.5^{'}$, we can measure ρ, θ. Additionally, the angle gauge device as shown Appendix B can measure the angle with respect to vertical to ground and thus becomes a measure of ϕ. Using these measurements, we define the camera position to take a picture and capture the images for target measured (ρ, θ, ϕ) values.

Based on this measurement procedure, a 50 image dataset as shown in Appendix C is created with computer screens displaying the ground truth values and filenames like *4p5_m45_120.jpg* to indicate $\rho = 4.5', \theta = -45°, \phi = 120°$.

4.2. Sample Output

In Figures 6 we show the sample image outputs from various stages of the image processing pipeline. When a sample image as shown in Figure 6a is fed through the Canny Edge Detector, we get an output with all linear and non-linear edges identified as shown in Figure 6b. As described before, this output is then taken to probabilistic Hough Line Transform and we get red lines as shown in Figure 6c. It must be mentioned that Hough Transform approach with a single setting is observed to not work for all scenarios, especially for very oblique scenarios such as $\theta = \pm 75°$. In this case it becomes imperative that the $minLineLength$ is reduced from 80 pixels to something smaller like 40 pixels. This indeed results in many more lines in the image that need to be processed by subsequent stages of pipeline but is essential to ensure smaller line edges in images for oblique scenarios are not missed out. Once the Hough line segments are detected, the quadrilateral detection algorithm is applied to determine the four edges of rectangular screen's image as a non-enclosing quadrilateral as in Figure 6c. Naturally when all 4 edges are established, four quadrilateral vertices are also determined. After the suitable quadrilateral is found, the centroid is simply the mean of all four vertices. Then the two vanishing points along length and width are found and the line of horizon joining them can be found. From the center of quadrilateral, a normal to this horizon line indicates the screen plane normal and is shown in Figure 6e as a red line. From the four edges of quadrilateral, the longest one is picked and accounting for it being width or length edge, a rectangle is overlaid on the quadrilateral as shown in Figure 6f where a green rectangle is overlaid on the red quadrilateral and the homography that makes this transformation is determined in the software implementation. After running through the ULDH algorithm via homography estimation, UL decomposition, setting up pair of non-linear equations and candidate solution, disambiguation via vanishing points, the final values of $\hat{\theta}$ and $\hat{\phi}$ are fully determined and are annotated as a sample functionality in Figure 6f. In addition to angle estimation, the distance heuristics are performed using quadrilateral area ratio and QA metric as defined before is calculated. For this sample, $\hat{\theta}$ and $\hat{\phi}$ are estimated as $43.2°$ and $119.1°$ while QA is 0.24. Since angle estimates are within the acceptable scene angle ranges defined earlier as, $20° < |\theta| < 70°$ and $55° < \phi < 125°$, and QA is above cut-off threshold of 0.075, this camera setting is declared *Good* scene setting, as annotated on image in Figure 6f.

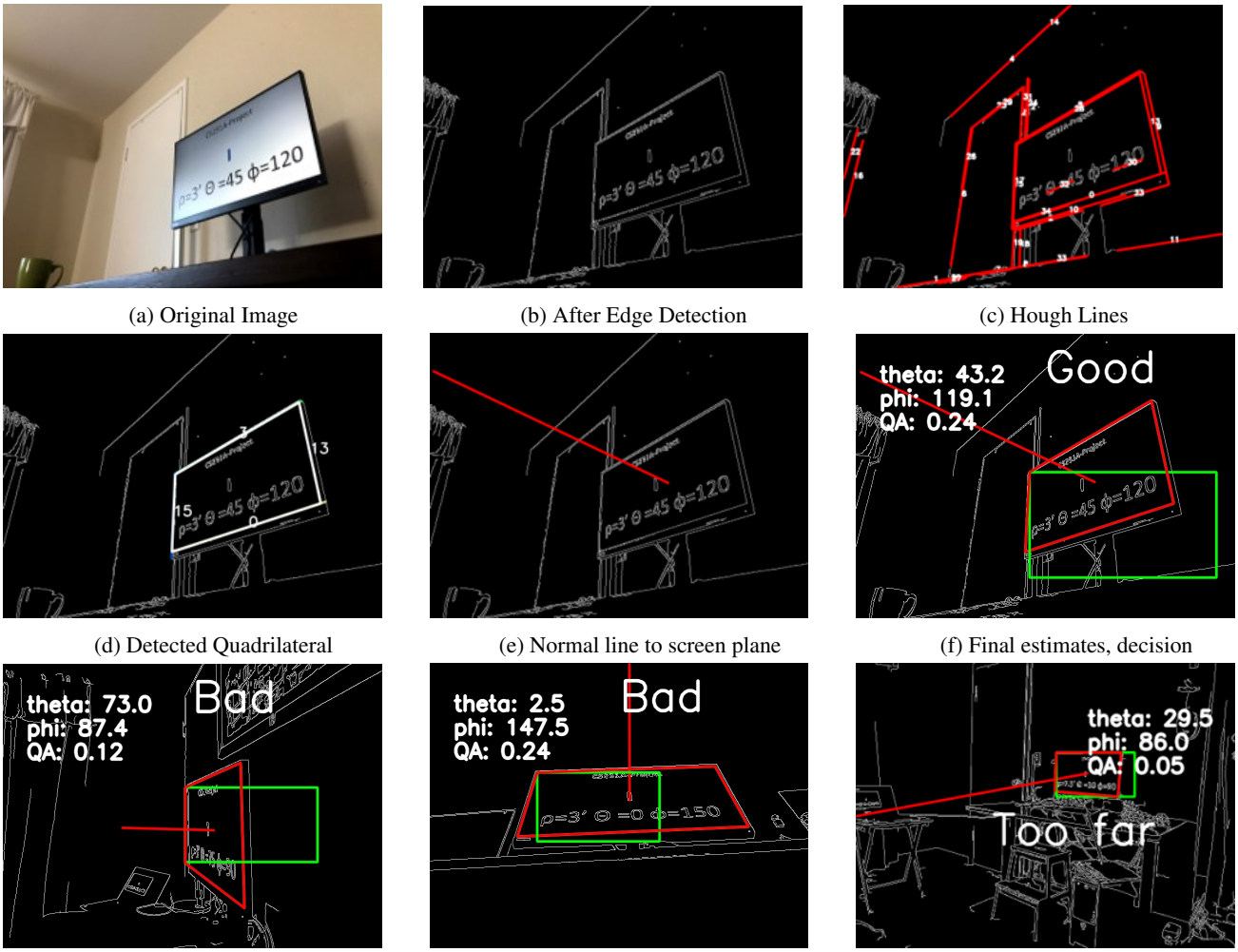
Metric	$\rho = 3'$	$\rho = 4.5'$	**Overall**
Samples	29	22	51
Mean Error	$3.26°$	$3.59°$	$3.41°$
Std. Dev.	$2.16°$	$2.47°$	$2.28°$
Min. Error	$0°$	$0°$	$0°$
Max. Error	$8.4°$	$10.4°$	$10.4°$
Misclassification	1	0	1

Table 1: Accuracy metrics for estimate $\hat{\theta}$ using ULDH

Figure 6 further shows some other sample outputs for other scene angle and distance settings. For instance, Figure 6g, shows a camera setting that is declared *Bad* based on θ estimate to be above the threshold for acceptable polar angle values. On the other hand, Figure 6h fails the passing criterion based on the threshold for azimuthal angle. Finally, Figure 6i shows a sample scenario rejected based on the screen being too far from the side camera because the QA values comes at 0.05 (below 0.075 threshold) and even though estimates $(\hat{\theta}, \hat{\phi}) = (29.5°, 86°)$ are valid from angle criterion. We can thus clearly see that the proposed method is able in these sample scenarios to discriminate good camera settings from bad ones based on the angle estimation and distance heuristics.

4.3. Evaluation

In order to validate the accuracy of ULDH method more exhaustively, we run the implemented software against the dataset collected as described in section 4.1. Figures 7a and 7b show the estimated angles versus the ground truth angles



(a) Original Image (b) After Edge Detection (c) Hough Lines

(d) Detected Quadrilateral (e) Normal line to screen plane (f) Final estimates, decision

(g) Rejected on θ threshold, GT:$\{\rho = 3', \theta = 75°, \phi = 90°\}$

(h) Rejected on ϕ threshold, GT:$\{\rho = 3', \theta = 0°, \phi = 150°\}$

(i) Rejected on distance heuristic, GT:$\{\rho = 7.5', \theta = 30°, \phi = 90°\}$

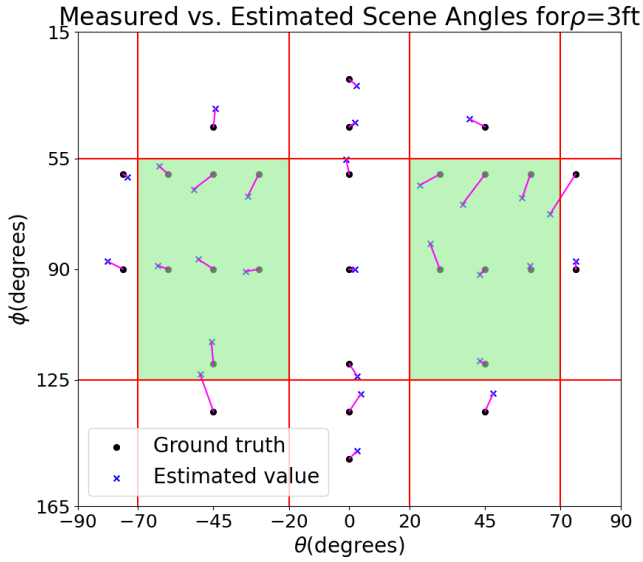
Figure 6: Sample Outputs for Various Stages/Scenarios

Metric	$\rho = 3'$	$\rho = 4.5'$	**Overall**
Samples	29	22	51
Mean Error	4.22°	3.5°	3.91°
Std. Dev.	3.31°	3.51°	3.38°
Min. Error	0°	0.1°	0°
Max. Error	12.5°	10.3°	12.5°
Misclassification	1	0	1

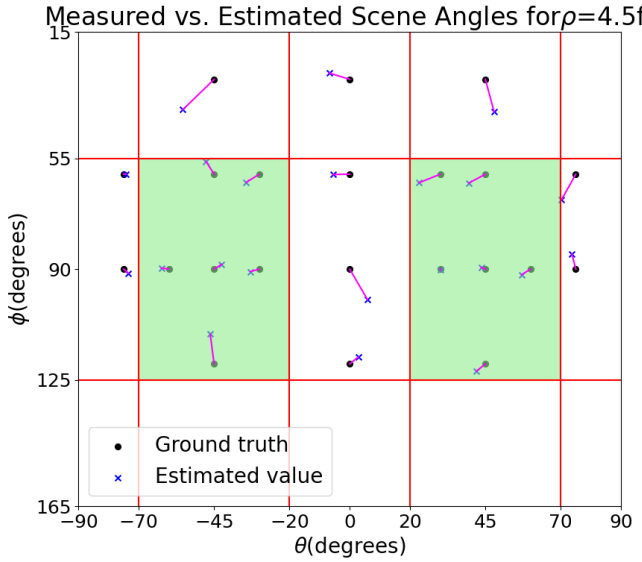
Table 2: Accuracy metrics for estimate $\hat{\phi}$ using ULDH

on $\theta - \phi$ plane for two settings of $\rho = 3'$ and $\rho = 4.5'$. As can be seen that, in general, the angle estimations using ULDH method are very close to the ground truth. On average, the ULDH method yields mean measurement error of $3.41°$ for θ and $3.91°$ for ϕ. All these statistics along with all the measured angles are shown in Table 1, Table 2 and Appendix E. Quite notably, even in the worst case, the maximum error observed for any of the angles is $12.5°$.

In terms of our original problem of classifying side camera settings based on angle estimate, we observe in Figure 7a that out of 29 data points for $\rho = 3'$, 2 samples are wrongly classified because estimation errors are enough to cause two images corresponding to $(\theta, \phi) = (75°, 60°)$ and $(\theta, \phi) = (-45°, 135°)$ misclassified as *Good* settings because angle estimates for these two points are moved into the valid (green rectangular areas) region for good camera settings due to estimation errors. On the other hand, none of the $\rho = 4.5$ angle estimations are erroneous enough to cause misclassification. Of the two misclassifications, one is caused by large enough error in $\hat{\theta}$ and one by that in $\hat{\phi}$. Based on overall 51 results, we report a 96% (49/51) classification



(a) Polar/Azimuthal Angles for $\rho = 3'$



(b) Polar/Azimuthal Angles for $\rho = 4.5'$

Figure 7: Overall results of angular estimation compared to ground truth (green regions indicate *Good* camera angles)

accuracy.

5. Summary/Further Work

In this work, a deeper understanding of projective geometry and planar homographies is developed and applied towards suitably decomposing the homography matrix to extract the scene angles. Further, by leveraging OpenCV's edge/line detectors, homography estimator and Python's non-linear equation solver, a complete image processing pipeline is built for camera setting classification for online exams. A key contribution of this work is in recognizing the upper-lower decomposability of homogaphy and efficiently using the resulting lower triangular matrix relations for scene angle estimation. By using this approach, mean estimation errors in both polar and azimuthal angles are less than 5° and bad camera placement are identified from good ones with greater than 95% accuracy.

Future work on this is anticipated to be in two directions. First, the ground truth data in this work was collected by aligning multiple off-the-shelf rulers/protractors/angle-gauges with greater likelihood of human-induced errors. It is thus imperative that more precise ground truth data be collected using better calibrated equipment so that the efficacy of newly proposed ULDH approach can be ascertained with even greater confidence. The other domain that this work will benefit from is the initial edge/line/quadrilateral detection pipeline that often breaks down, especially for very oblique angles, and requires changes in settings and/or greater computation needs. This can be improved by either convex contour detections or CNN based object detectors.

References

[1] *Online Cheating Amid COVID-19*, E. Bilen, A. Matros Journal of Economic Behavior and Organization, vol. 182, pp. 196-211, Feb. 2021.

[2] *NSB Info for Coaches* , ref. p6.

[3] *A Computational Approach to Edge Detection*, J. Canny, IEEE Transactions and Pattern Matching and Machine Intelligenc, pp. 679-698, Nov. 1986.

[4] *Progressive probabilistic Hough transform for line detection*, C. Galamhos; J. Matas; J. Kittler, Proc. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition

[5] *Motion and structure from motion in a piecewise planarenvironment*, O. Faugeras, F. Lustman., International Journal of Pattern Recognition and Artificial Intelligence, pp483-508, Sept. 1988.

[6] *Deeper understanding of the homography decomposition for vision-based control*, E.Malis, M. Vargas, INRIA Report, Jan. 2007.

[7] *Estimation of relative camera positions for uncalibrated cameras*, R.Hartley, European Conference on Computer Vision, pp.579-587, 1992.

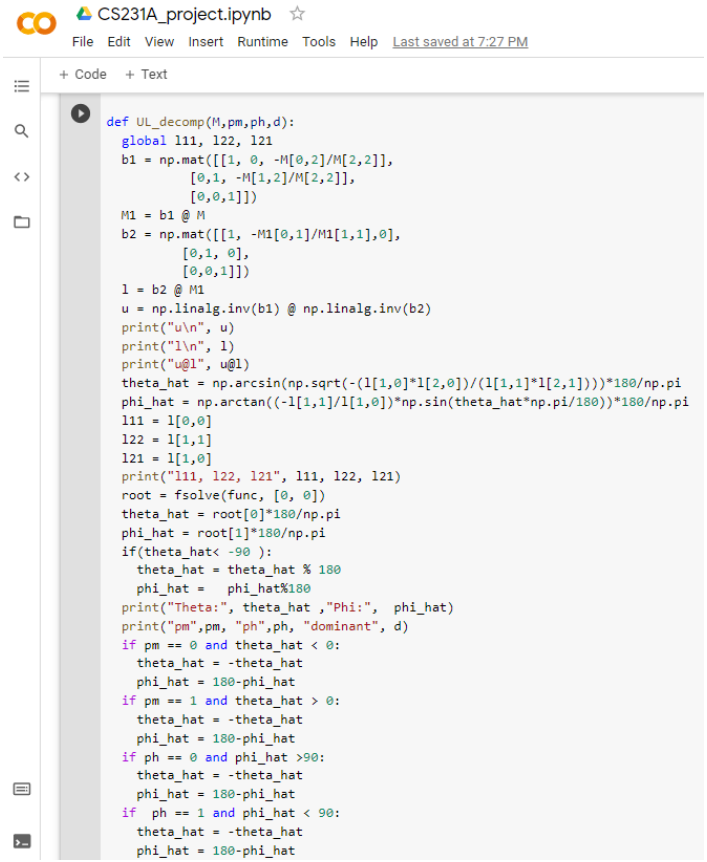
[8] *Real-time estimation of 3D scene geometry from a single image*, C. Jung, C. Kim, Elsevier Journal on Pattern Recog., vol 45, Issue 9, pp. 3256-3269, Sept. 2012

[9] *3D Viewing and Projection Transformations*, R. Eckert , Online.

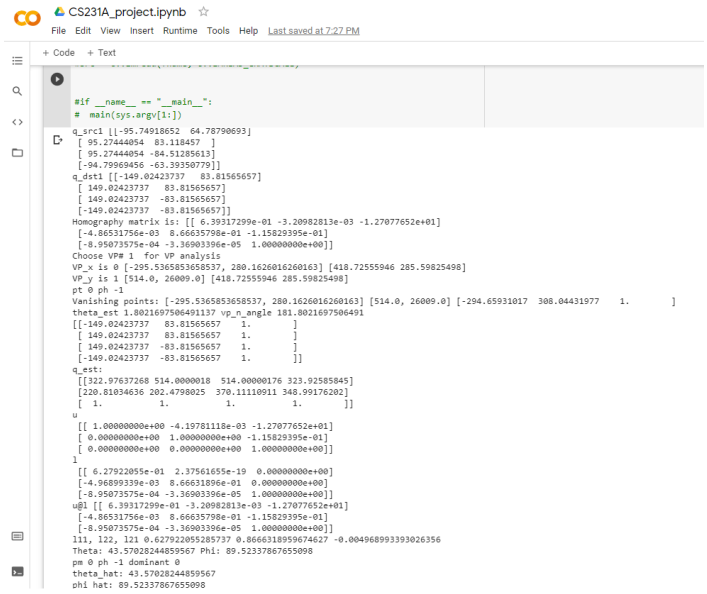
[10] OpenCV, *decomposeHomogaphyMat*, Canny edge detector, *Probabilistic Hough Line Transform*.

A. Sample Code Snippets

Link to Python notebook on Github.



(a) Sample Code (UL decomposition)



(b) Sample Log message output

B. Measurement Tools and Procedure

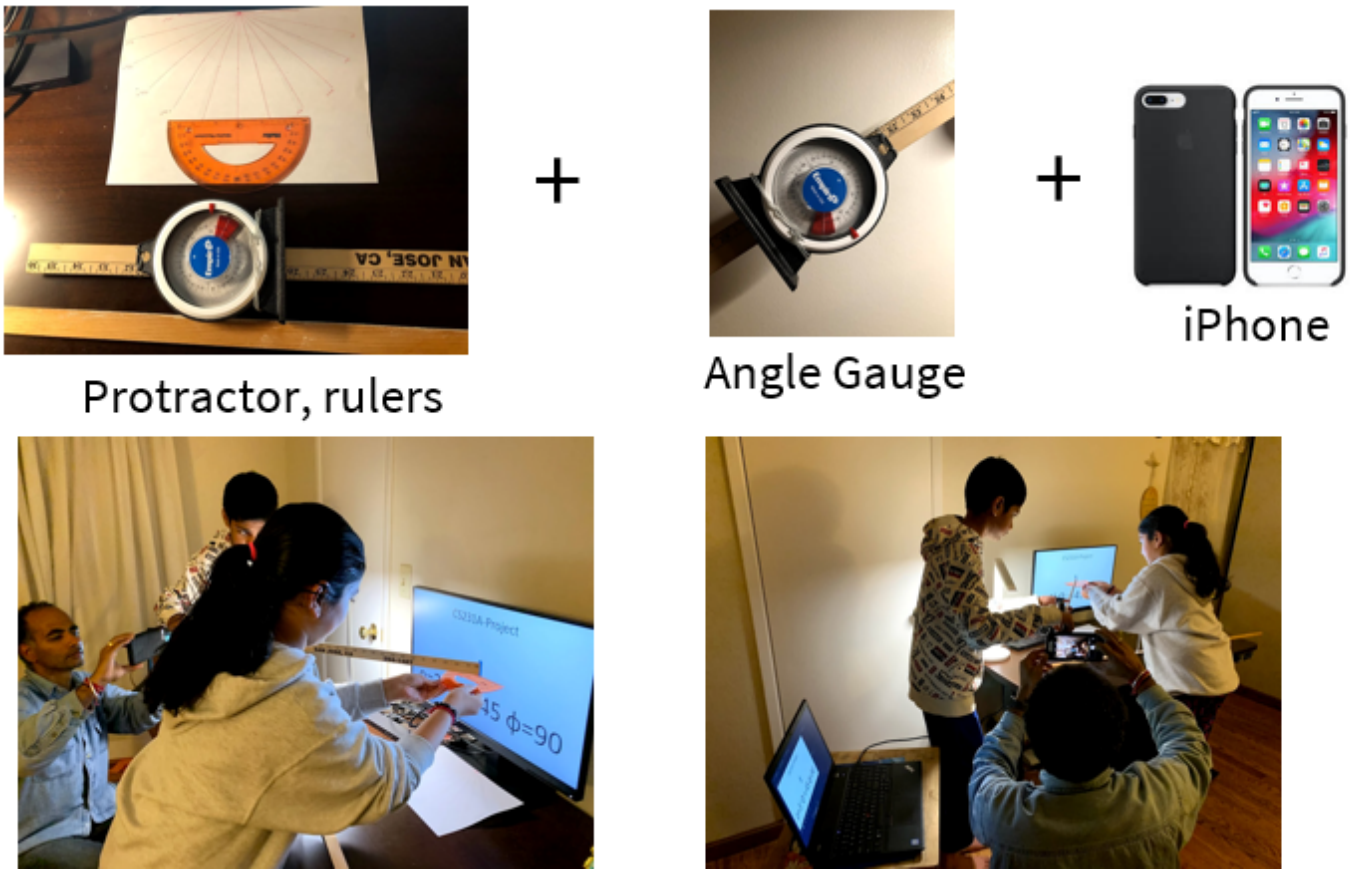


Figure 9: Tools/procedures for capturing image dataset with ground truth

C. 50-Image Dataset with ρ, θ, ϕ Ground Truth Values

Image Dataset

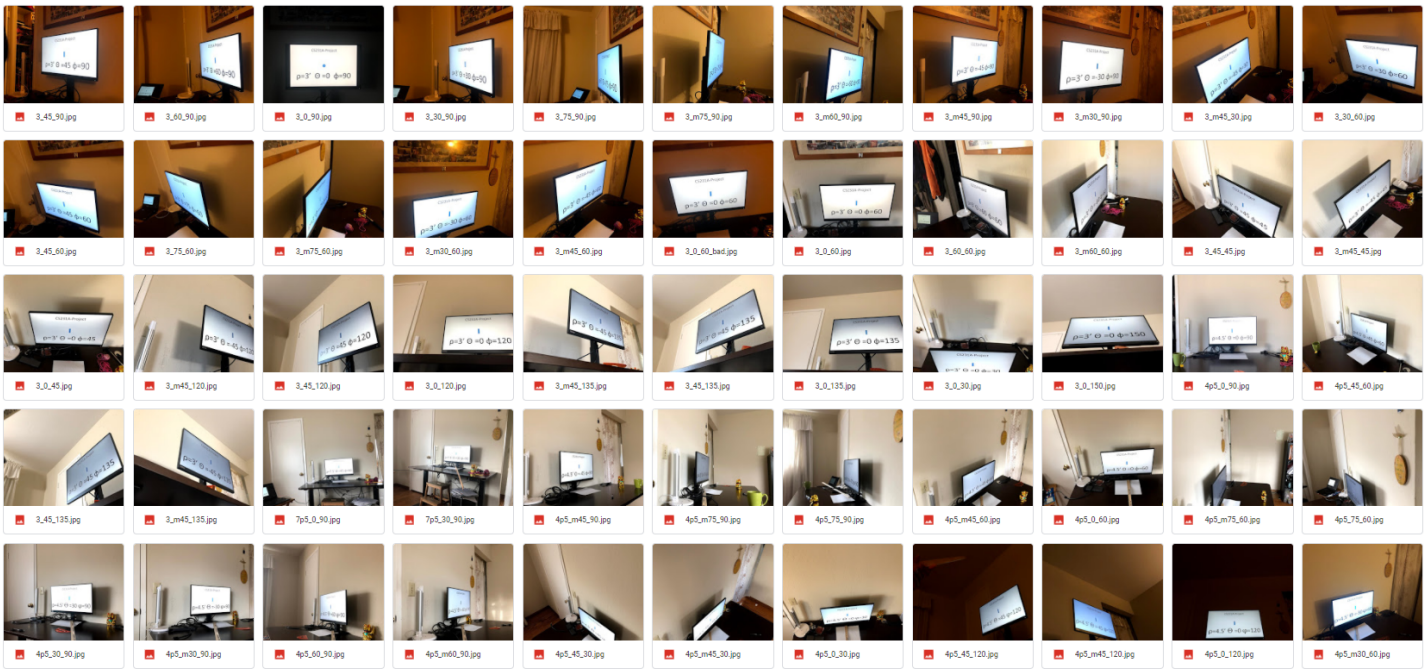


Figure 10: Image dataset with ground truth distance, angles

D. Image processing results with estimated angles and decision

Output images

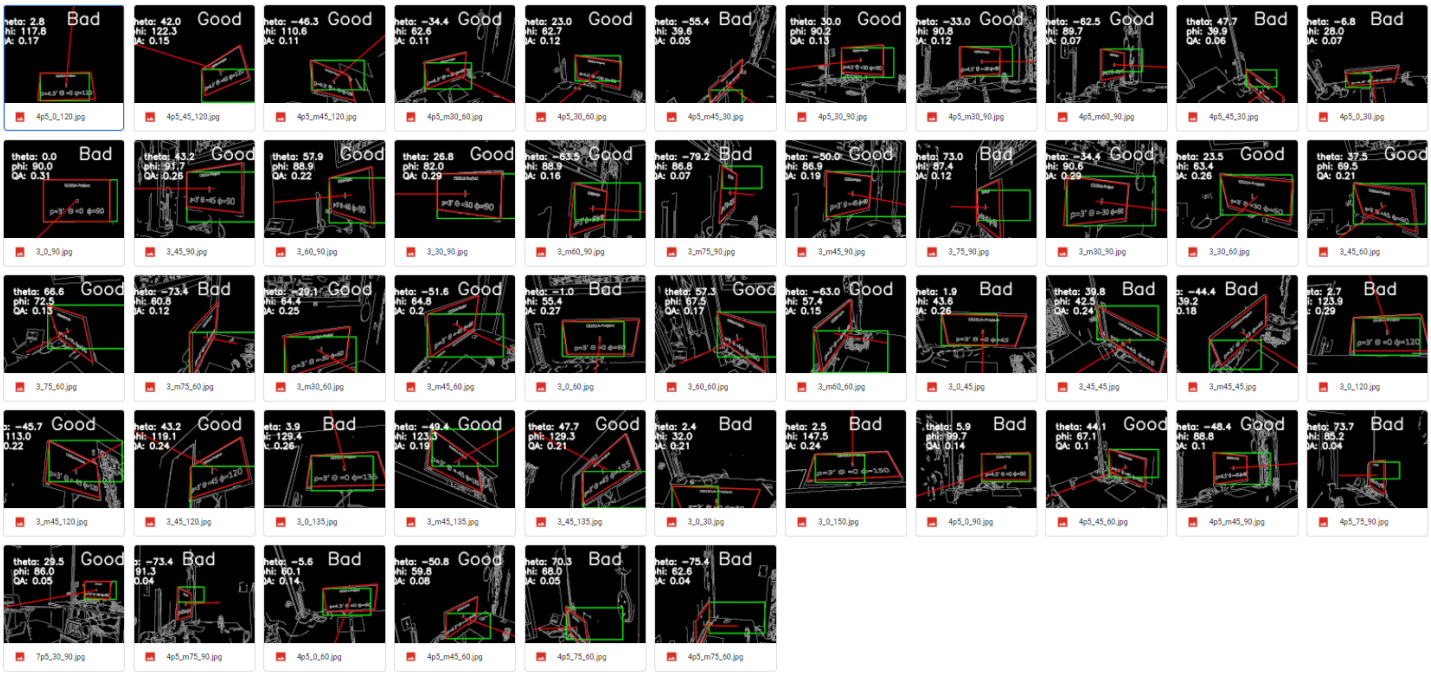


Figure 11: Processed image results

E. Angle Measurements for Complete Dataset

Rho	Theta	Phi	Theta_hat	Phi_hat	Theta_delta	Phi_delta	Abs. theta error	Abs Phi error	GT -decision	EST. Decision	Match?
3	30	60	23.5	63.4	6.5	-3.4	6.5	3.4	Good	Good	Match
3	45	60	37.5	69.5	7.5	-9.5	7.5	9.5	Good	Good	Match
3	75	60	66.6	72.5	8.4	-12.5	8.4	12.5	Bad	Good	Mismatch
3	-75	60	-73.4	60.8	-1.6	-0.8	1.6	0.8	Bad	Bad	Match
3	-30	60	-33.6	66.9	3.6	-6.9	3.6	6.9	Good	Good	Match
3	-45	60	-51.6	64.8	6.6	-4.8	6.6	4.8	Good	Good	Match
3	0	90	1.9	90	-1.9	0	1.9	0	Bad	Bad	Match
3	45	90	43.2	91.7	1.8	-1.7	1.8	1.7	Good	Good	Match
3	60	90	59.8	89	0.2	1	0.2	1	Good	Good	Match
3	30	90	26.8	82	3.2	8	3.2	8	Good	Good	Match
3	-60	90	-63.5	88.9	3.5	1.1	3.5	1.1	Good	Good	Match
3	-75	90	-80	87.6	5	2.4	5	2.4	Bad	Bad	Match
3	-45	90	-50	86.9	5	3.1	5	3.1	Good	Good	Match
3	75	90	75	87.4	0	2.6	0	2.6	Bad	Bad	Match
3	-30	90	-34.4	90.8	4.4	-0.8	4.4	0.8	Good	Good	Match
3	0	60	-1	55.4	1	4.6	1	4.6	Bad	Bad	Match
3	60	60	57.3	67.5	2.7	-7.5	2.7	7.5	Good	Good	Match
3	-60	60	-63	57.4	3	2.6	3	2.6	Good	Good	Match
3	0	45	1.9	43.6	-1.9	1.4	1.9	1.4	Bad	Bad	Match
3	45	45	39.8	42.5	5.2	2.5	5.2	2.5	Bad	Bad	Match
3	-45	45	-44.4	39.2	-0.6	5.8	0.6	5.8	Bad	Bad	Match
3	0	120	2.7	123.9	-2.7	-3.9	2.7	3.9	Bad	Bad	Match
3	-45	120	-45.7	113	0.7	7	0.7	7	Good	Good	Match
3	45	120	43.2	119.1	1.8	0.9	1.8	0.9	Good	Good	Match
3	-45	135	-49.4	123.3	4.4	11.7	4.4	11.7	Bad	Good	Mismatch
3	0	135	3.9	129.4	-3.9	5.6	3.9	5.6	Bad	Bad	Match
3	45	135	47.7	129.3	-2.7	5.7	2.7	5.7	Bad	Bad	Match
3	0	30	2.4	32	-2.4	-2	2.4	2	Bad	Bad	Match
3	0	150	2.5	147.5	-2.5	2.5	2.5	2.5	Bad	Bad	Match
4.5	0	90	5.9	99.7	-5.9	-9.7	5.9	9.7	Bad	Bad	Match
4.5	45	60	39.5	62.8	5.5	-2.8	5.5	2.8	Good	Good	Match
4.5	-45	90	-42.5	88.5	-2.5	1.5	2.5	1.5	Good	Good	Match
4.5	75	90	73.7	85.2	1.3	4.8	1.3	4.8	Bad	Bad	Match
4.5	-75	90	-73.4	91.3	-1.6	-1.3	1.6	1.3	Bad	Bad	Match
4.5	0	60	-5.6	60.1	5.6	-0.1	5.6	0.1	Bad	Bad	Match
4.5	-45	60	-47.7	55.9	2.7	4.1	2.7	4.1	Good	Good	Match
4.5	75	60	70.3	68	4.7	-8	4.7	8	Bad	Bad	Match
4.5	-75	60	-74.2	59.9	-0.8	0.1	0.8	0.1	Bad	Bad	Match
4.5	30	90	30	90.2	0	-0.2	0	0.2	Good	Good	Match
4.5	-30	90	-33	90.8	3	-0.8	3	0.8	Good	Good	Match
4.5	60	90	57	91.8	3	-1.8	3	1.8	Good	Good	Match
4.5	-60	90	-62.5	89.7	2.5	0.3	2.5	0.3	Good	Good	Match
4.5	45	30	47.9	40.3	-2.9	-10.3	2.9	10.3	Bad	Bad	Match
4.5	0	30	-6.8	28	6.8	2	6.8	2	Bad	Bad	Match
4.5	-45	30	-55.4	39.6	10.4	-9.6	10.4	9.6	Bad	Bad	Match
4.5	0	120	2.8	117.8	-2.8	2.2	2.8	2.2	Bad	Bad	Match
4.5	45	120	42	122.3	3	-2.3	3	2.3	Good	Good	Match
4.5	-45	120	-46.3	110.6	1.3	9.4	1.3	9.4	Good	Good	Match
4.5	-30	60	-34.4	62.6	4.4	-2.6	4.4	2.6	Good	Good	Match
4.5	30	60	23	62.7	7	-2.7	7	2.7	Good	Good	Match
4.5	45	90	43.6	89.5	1.4	0.5	1.4	0.5	Good	Good	Match
7.5	30	90	29.5	86	0.5	4	0.5	4	Good	Good	Match
						MEAN	3.41	3.91			
						STDEV	2.28	3.38			
						MAX	10.40	12.50			
						MIN	0.00	0.00			

Figure 12: Angular Estimation Results versus Ground Truth