# Theory of Ground Plane Estimation from Sequential Imaging

Thomas W. C. Carlson

4-20-2021

**Abstract**

This paper presents an approach to the determination of a ground (or dominant) plane from a sequence of two images using theoretical data. The two images are assumed to have many identical points in them due to their temporal proximity, and from these corresponding points a ground plane homography and optical flow analysis is conducted. The degree by which these two methods differ drives the classification of a feature point as "belonging" to the ground plane of the image or not.

## 1 Introduction

In mobile robotics, a topic of key importance is safe navigation within the real world. A number of problems arise that must be considered: wall location, motion tracking and estimation, and stability of the ground plane. Errors in the determination of safety in any of these areas can lead to an autonomous robot making poor decisions which may result in severe and costly failures.

This paper aims to address the problem of identifying passable terrain by identifying the "ground" plane in the robot's camera view of the real world. First, a selection of key feature points to be tracked between two images is made. From this set of key features that can be found in both images, a selection is made based on some reasonable assumptions about what the ground plane region should look like from the camera perspective (keeping in mind that this is application-dependent). From this set and its corresponding points in a second image, an assumed ground plane homography is calculated. This homography is then applied to every key feature found in the image to locate the feature's future position if it belonged to the ground plane. This is then compared to data extracted from optical flow between the same point correspondences in the same two images. The difference between the two is treated as an error, and errors above a certain threshold indicate that the specified feature does not lie on the ground plane. Connecting all inlier points which are found to belong to the ground plane produces a mesh of the traversible area for the robot to employ in its pathing calculations.

As a method of verification of these techniques, an image of a hallway with a tiled floor (so as to include many candidate feature points on the ground plane) was used in the individual components of this project, shown in Figure 1. Tracing the image to simplify and bold key features was done to ensure consistent results, as shown in Figure 2. However, in the face of significant technological and hardware difficulties, these techniques could not be implemented over the course of a semester on real data with a real robot, and so this work relies on [5] to demonstrate the effectiveness of this method.

Figure 1: Image of a hallway through which a robot might pass.

## 2 Feature Identification

A number of ways have been identified in literature which govern the selection of features that are easy to recognize across multiple images. One of the most common implementations of this is the Harris Corner Detector algorithm.

A corner is simply defined for image processing purposes by the edges that make it up. The process for edge detection is well understood and has been covered in lecture. It involves investigation of the gradient changes across an image, or a directional derivative of the image in the horizontal ($I_x$) or vertical ($I_y$) directions.

Edges and corners can also be characterized as regions of an image in which a large change can be felt when a window of that region is shifted slightly. To that end, the Harris Corner detection algorithm seeks to maximize the equation

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{1}$$

where E is the error value, or difference between the image windows before and after shifting. The displacement in x and y are given by u and v respectively. w(x,y) is a mask which represents the window being used, and I(x+u,y+v) and I(x,y) are the intensities of the images after and before being moved, respectively.

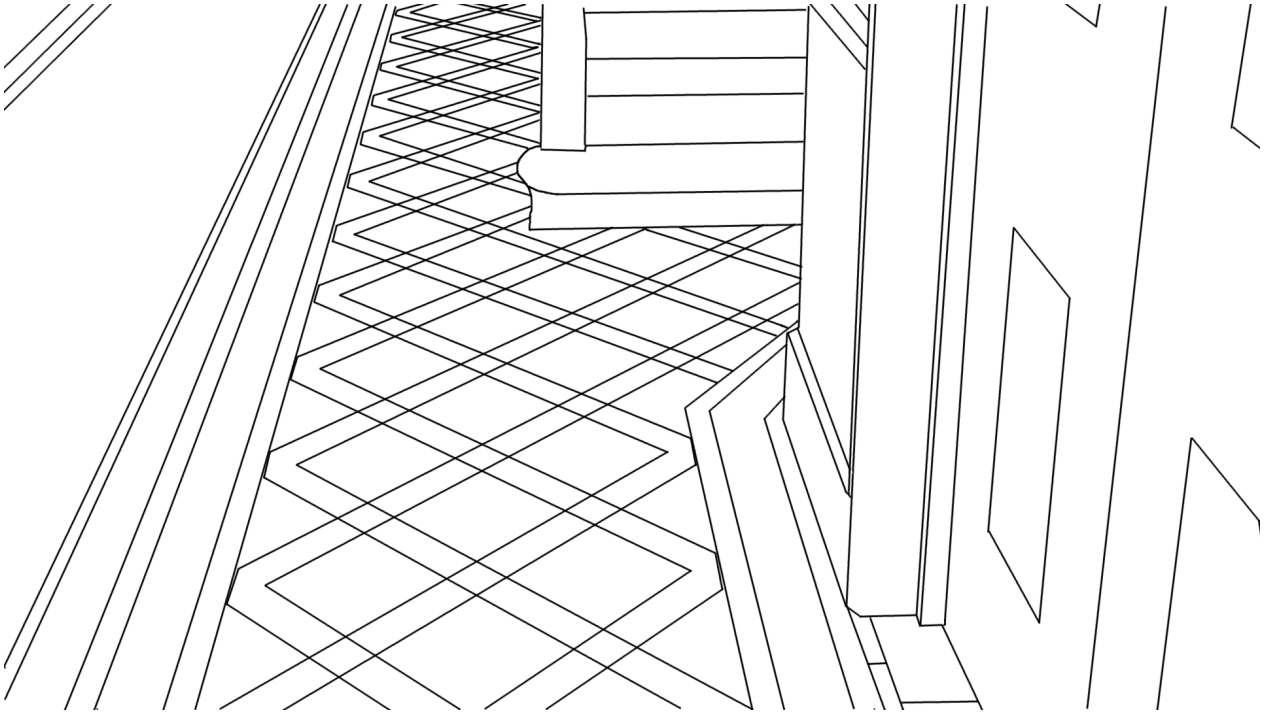

Figure 2: Traced version of the hallway to make corner identification easier.

Per the previous discussion, high values of E indicate a very clear corner, so the algorithm will seek maximal values of E.

By performing a first-order Taylor series expansion, and reorganizing the terms to be a matrix equation, we can find

$$E(u,v) \approx \sum_{x,y} u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2 \tag{2}$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left( \sum w(x,y) \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix} \tag{3}$$

We can then classify the "corner response" using the eigenvalues $(\lambda_1, \lambda_2)$ of the summation matrix. If there is little to no difference between the eigenvalues and they are small, then the error is almost constant in every direction. If they are both large, then E increases in all directions. When one is significantly larger than the other, the corner is actually just a single edge. This classification can be obtained using a corner response equation:

$$R = detM - k(traceM)^2 \tag{4}$$

$$detM = \lambda_1\lambda_2 \tag{5}$$

$$traceM = \lambda_1 + \lambda_2 \tag{6}$$

When $R$ is large, we find a corner. When $R$ is small, the region is flat, or featureless. When $R$ is negative and of large magnitude, the region contains an edge. By setting a threshold for R, we can decide whether the region contains a corner. The complete process of locating corners in an image is captured by Algorithm 1.

**Algorithm 1:** Harris Corner Detection Algorithm

**Result:** Locate all "corners" in an images
Compute the x and y derivatives of the image
Compute products of the derivatives at each pixel
Compute the sums of the products of the derivatives at each pixel
Calculate the eigenvalues of the resulting matrix
Calculate the response $R$
Apply a threshold to $R$ values and suppress non-maximal values

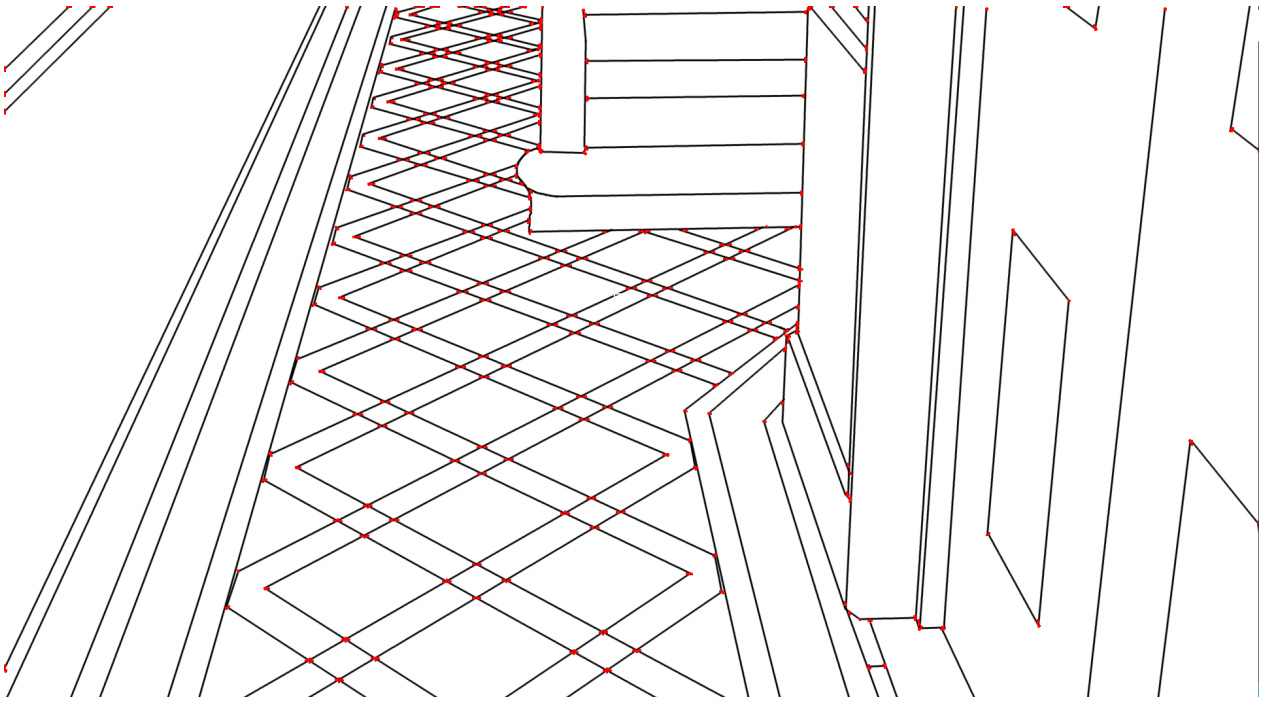

Figure 3: Harris corner detection has been applied to the image.

This has been implemented in many image processing libraries and is one of the most fundamental feature extraction techniques in image processing. Returning to the test image of Figure 2, using OpenCV's .cornerHarris() function produces the output in Figure 3. The programmatic Python implementation is discussed in [1].

# 3 Obtaining the Ground Plane Homography

The ground plane homography is an expression of the mapping of a point on the ground plane in the first image, $I_n$, to its corresponding position in the second image, $I_{n+1}$. The homography combines the geometry of the world coordinates, in which the camera resides, and the two images to calculate the difference in angle and position of the projected image taken by the camera. In this way, the homography is also a transformation between the two image planes.

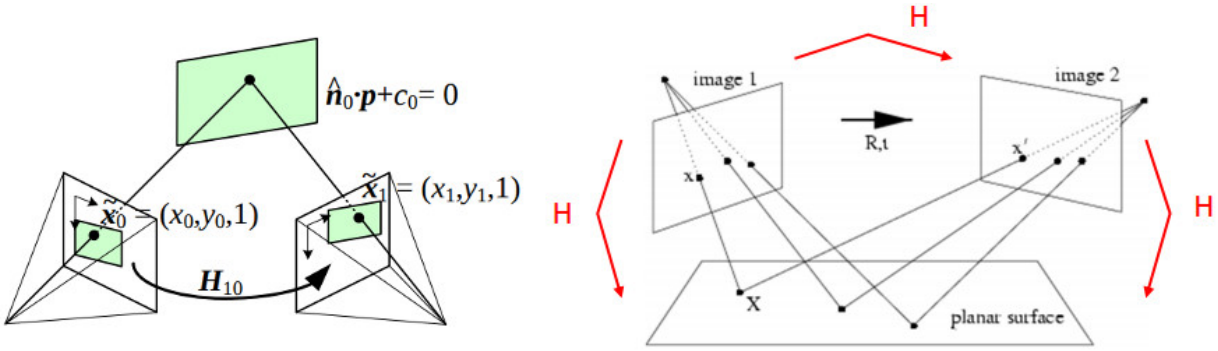


Figure 4: Homographies can relate different images of the same surface.

Typically, the homography is expressed as a $3x3$ matrix with 8 degrees of freedom as in equation (7). It can be used both to relate the world to the image plane as well as different image planes, as will be used in this paper and is shown in Figure 4.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7}$$

In order to calculate the homography it is important to first identify key features in the images which are shared. Their positions on each image can then be determined and used to calculate the homography matrix. Seeking the homography amounts to solving equation (8), in which the point $(x_i, y_i)$ belongs to the source image and the point $(x'_i, y'_i)$ belongs to the destination image, and $H$ is the homography. In order to find $H$, all 8 degrees of freedom must be constrained and thus at least 4 corresponding (and therefore 8 points in total) must be used. $s_i$ is a scale factor, which indicates that the homography is defined up to scale, and this factor can typically be ignored for calculation purposes.

$$s_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \approx H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{8}$$

As an example derivation via the method from [2], consider two rectangles as defined by their corner points

$$A = (p_{1Ax}, p_{1Ay}), (p_{2Ax}, p_{2Ay}), (p_{3Ax}, p_{3Ay}), (p_{4Ax}, p_{4Ay}) \tag{9}$$
$$B = (p_{1Bx}, p_{1By}), (p_{2Bx}, p_{2By}), (p_{3Bx}, p_{3By}), (p_{4Bx}, p_{4By}) \tag{10}$$

Where corner $1A$ corresponds to corner $1B$ and the goal is to calculate a homography matrix $H_A^B$ which transforms points in $A$ to points in $B$. These correspondences can be expressed as a set of 4 2x9 matrices by manipulating the homography matrix:

$$\begin{bmatrix} x_{iB} \\ y_{iB} \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11}x_{iA} + h_{12}y_{iA} + h_{13} \\ h_{21}x_{iA} + h_{22}y_{iA} + h_{23} \\ h_{31}x_{iA} + h_{32}y_{iA} + h_{33} \end{bmatrix} = \begin{bmatrix} \frac{h_{11}x_{iA}+h_{12}y_{iA}+h_{13}}{h_{31}x_{iA}+h_{32}y_{iA}+h_{33}} \\ \frac{h_{21}x_{iA}+h_{22}y_{iA}+h_{23}}{h_{31}x_{iA}+h_{32}y_{iA}+h_{33}} \\ 1 \end{bmatrix} \tag{11}$$

$$(h_{31}x_{iA} + h_{32}y_{iA} + h_{33})x_{iB} = h_{11}x_{iA} + h_{12}y_{iA} + h_{13} \tag{12}$$

$$(h_{31}x_{iA} + h_{32}y_{iA} + h_{33})y_{iB} = h_{21}x_{iA} + h_{22}y_{iA} + h_{23} \tag{13}$$

$$p_i = \begin{bmatrix} -x_{iA} & -y_{iA} & -1 & 0 & 0 & 0 & x_{iA}x_{iB} & y_{iA}x_{iB} & x_{iB} \\ 0 & 0 & 0 & -x_{iA} & -y_{iA} & -1 & x_{iA}y_{iB} & y_{iA}y_{iB} & y_{iB} \end{bmatrix} \tag{14}$$

which can be stacked into a matrix $P$, while $H$ is remapped as

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \rightarrow \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = H' \tag{15}$$

The task is now reduced to solving

$$PH' = 0 \tag{16}$$

so long as the additional constraint of $h_9 = 1$ is applied and the trivial solution of $H' = 0$ is discarded, giving the full problem as

$$PH' = \begin{bmatrix} -x_{1A} & -y_{1A} & -1 & 0 & 0 & 0 & x_{1A}x_{1B} & y_{1A}x_{1B} & x_{1B} \\ 0 & 0 & 0 & -x_{1A} & -y_{1A} & -1 & x_{1A}y_{1B} & y_{1A}y_{1B} & y_{1B} \\ -x_{2A} & -y_{2A} & -1 & 0 & 0 & 0 & x_{2A}x_{2B} & y_{2A}x_{2B} & x_{2B} \\ 0 & 0 & 0 & -x_{2A} & -y_{2A} & -1 & x_{2A}y_{2B} & y_{2A}y_{2B} & y_{2B} \\ -x_{3A} & -y_{3A} & -1 & 0 & 0 & 0 & x_{3A}x_{3B} & y_{3A}x_{3B} & x_{3B} \\ 0 & 0 & 0 & -x_{3A} & -y_{3A} & -1 & x_{3A}y_{3B} & y_{3A}y_{3B} & y_{3B} \\ -x_{4A} & -y_{4A} & -1 & 0 & 0 & 0 & x_{4A}x_{4B} & y_{4A}x_{4B} & x_{4B} \\ 0 & 0 & 0 & -x_{4A} & -y_{4A} & -1 & x_{4A}y_{4B} & y_{4A}y_{4B} & y_{4B} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{17}$$

which can be solved by pre-multiplication of the expression by $P^{-1}$. When there are more than 4 points to consider for this calculation it can be deemed overdetermined, and it becomes best to use software to perform a linear least squares fit. This is done via Singular Value Decomposition, and the "best" fit can be found directly to yield $H$. The procedure is widely implemented in OpenCV and Matlab:

$$P = U\Sigma V^T = \sum_{i=1}^{9} \sigma_i u_i v_i^T \tag{18}$$

where the "right singular vector" from $V$ corresponding to the smallest singular value $\sigma_9$ (the best fit, as $\sigma_i$ are returned in descending order) is $H'$ which can then be reshaped into $H$ as described in (15).

With this effort concluded, we now turn to the selection of features from the images to use as corresponding points to calculate the homography. Clearly we desire a homography mapping points on the ground plane in image $I_n$ to the ground plane in image $I_{n+1}$, and must restrict the selected features to ones which appear on the ground plane. The simplest way to do this, though not applicable in all situations, is to assume that a specified static area in front of the robot is always clear and contains points lying on the ground plane. This assumption is often valid in robotics as for the robot to have traveled in a direction in which this would not be valid would indicate that either a more specific objective has been identified which uses other navigation systems or significant errors were already present in the determination of what space is "free" for the robot to travel through.

To make the selection, a rectangle or trapezoid large enough to capture a swath of points in the space through which a robot could move is calculated and placed at the lower edge of the image. Only feature points which lie within this area are then considered for matching in the second image. From these matches the homography is computed.

# 4 Optical Flow Estimation

In order to check if any given point in an image belongs to the ground plane the objective is to check whether its movement between image $I_n$ and image $I_{n+1}$ conforms to the previously obtained homography in section 3. One way to do this is via optical flow, which seeks to follow the movement of image intensities across two images.

The basis of optical flow theory relies on the relative motion between objects in a scene and the camera capturing the scene. It is summarized in Figure 6 as the change in image intensity from $I$ to $I'$ as a function of changing pixel locations $x$, $y$ and time $t$. Initially, it makes sense to assume that very little change has taken place over the period of time $t$ (so the method applies best to slow-moving objects in a scene), and so

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{19}$$

Taking the Taylor Series approximation of the right hand side and dividing by $dt$ yields the optical flow equation

$$I(x + dx, y + dy, t + dt) \approx I(x, y, t) + I_x dx + I_y dy + I_t dt \tag{20}$$

$$I(x + dx, y + dy, t + dt) - I(x, y, t) = I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t \tag{21}$$

$$0 = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} \tag{22}$$

$$0 = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} \tag{23}$$

Given that we know the image $I$, we are faced with 2 unknown quantities in one equation: the change in $x$ ($u$), $y$ ($v$) over time. $t$ can be found from timestamping of the images, which is often easily obtained as this

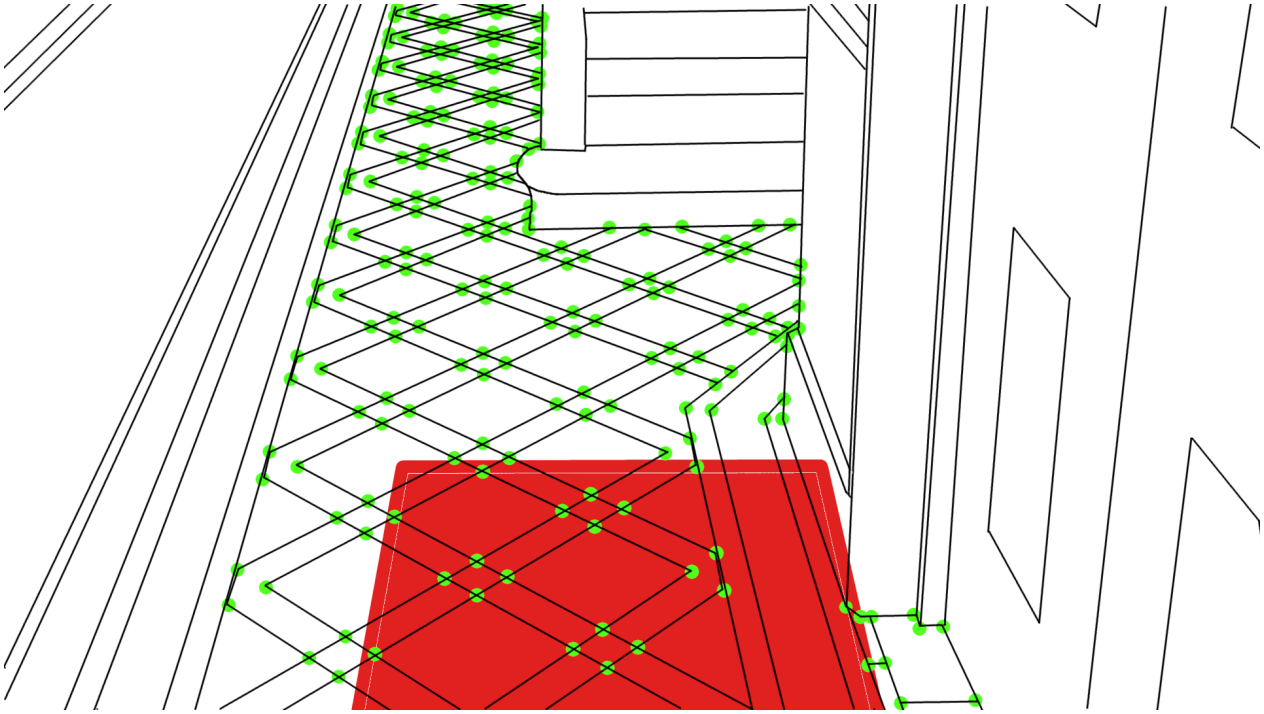

Figure 5: Feature points are marked in green for visualization, while the red region represents the selection region, which is application dependent and up to the operator. Note that the selection need not be perfect, and the region may be algorithmically restricted to increase the goodness of the fit in later parts of the ground plane identification procedure.

method is widely employed on video systems where the datastream must be timestamped regardless. The problem becomes the identification of change in the $x$ and $y$ directions. Due to the number of unknowns, the problem cannot be directly solved. Takeo Kanade and Bruce Lucas together proposed a solution to this problem in [3] by implementing a second assumption: that a pixel's neighbors typically move in the same directions without significantly departing from each other because objects in space are coherent. This assumption is often called "coherence," and forms the basis for many other image processing techniques such as Kanade-Lucas-Tomasi feature selection.

As a result, the use of a window is proposed, which captures pixel data for $n^2$ pixels for a window of size $n$. From each pixel an equation can be found of the form given above, resulting in an overdetermined problem with a set of many equations

$$\begin{aligned} I_x(p_1)u + I_y(p_1) + I_t &= 0 \\ I_x(p_2)u + I_y(p_2) + I_t &= 0 \\ &\vdots \\ I_x(p_m)u + I_y(p_m) + I_t &= 0 \end{aligned}$$

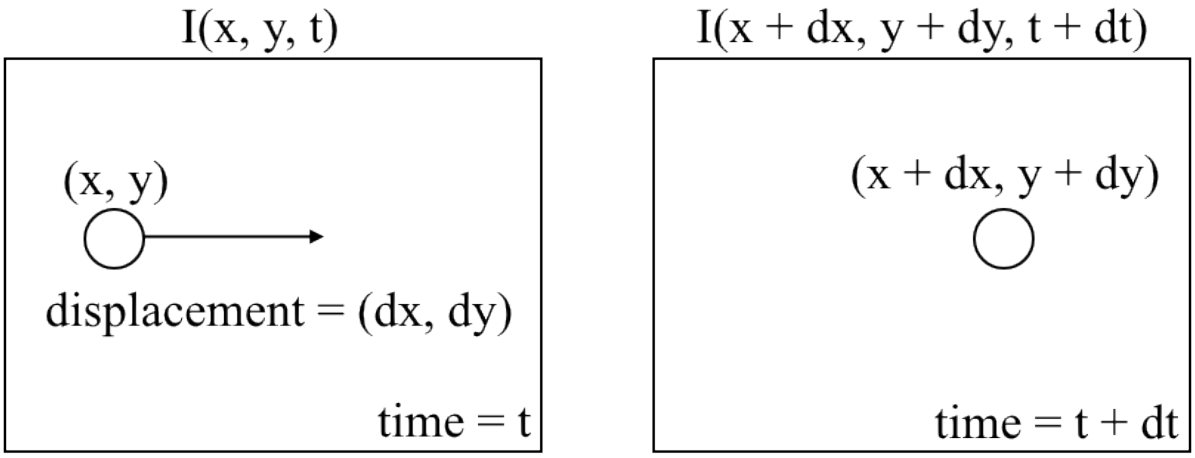


Figure 6: Optical Flow.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_m) & I_y(p_m) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \tag{24}$$

This is an equation of the form $Ad = b$ and can be solved via the linear least squares approach

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} \tag{25}$$

from which solutions can be found for the pixel at the center of every window. This is necessarily computationally expensive to perform on an entire image's worth of pixels, and so we again rely on feature detection for candidate points.

# 5 When Does a Feature Belong to the Ground Plane?

Given some set of candidate points from the feature tracking of section 2, and two ways of examining their positions in sequential images, it is now possible to estimate whether a given point is present on the same ground plane that can be found in both images in the sequence.

The process involves using the homography found in section 3 as a point of comparison for where a pixel "should" end up if it belonged to the ground plane, and then comparing that theoretical result to the actual result obtained using optical flow. The difference in these values can then be thresholded such that a lower value is treated as "no difference" or indeed belonging to the ground plane. Values above the threshold indicate that there is a significant difference between the expected ground plane homography transformation of the feature and the actual transformation witnessed by the camera, and thus the point does not belong to the ground plane. This has been summarized as Algorithm 2.

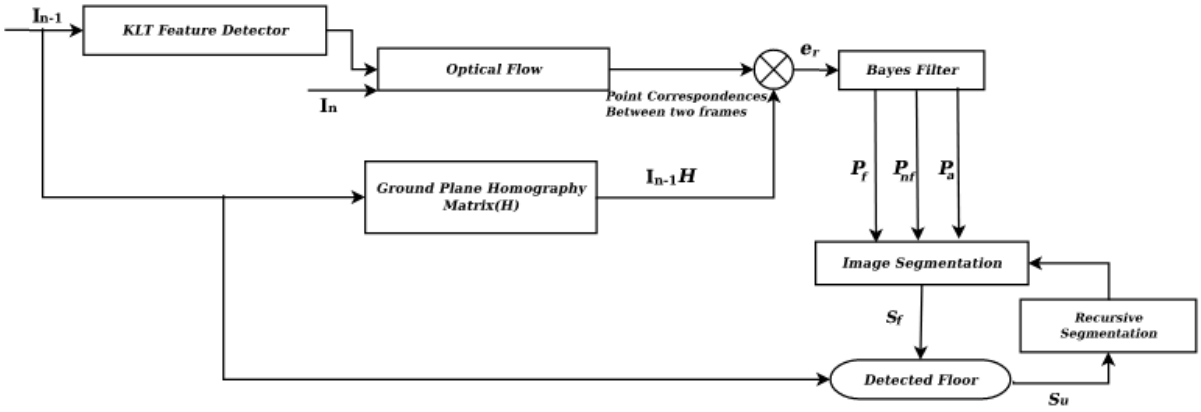


Figure 7: Ground Plane Segmentation flow chart. $e_r$ is the difference between the two calculated point locations, $P_f$ is the probability that a point belongs to the ground plane, $P_{nf}$ is the probability that a point does not belong the ground plane, and $P_a$ accounts for ambiguous cases.

This method has been demonstrated to work in literature, as in [5]. The authors employed a Bayesian decision network and recursive image segmentation to improve their results by refining the decision making process (the 'thresholding' discussed above) over successive calculations of a feature's belongingness to the ground plane and adjusting the Bayesian prior. The workflow of their methodology is given in Figure 7. Although this is not discussed in this paper, Figure 8 demonstrates that considerable improvements can be made by successfully updating the Bayesian decision making process with more information as the image sequence is extended.

**Algorithm 2:** Feature Point Classification Algorithm

**Result:** Decide whether a point belongs to the ground plane
Execute the Harris Corner Detection algorithm
Use the subset of feature points belonging to the basis estimate plane to calculate the ground plane homography
For every feature selected in the image, find the corresponding motion of that point experienced by optical flow
From the motion found by optical flow, calculate the final position of the feature
For the same feature, compute the location of the feature if it belongs to the ground plane using the ground plane homography
Calculate the difference between the two locations found
Apply a threshold to the difference to classify the feature as belonging to the ground plane or otherwise

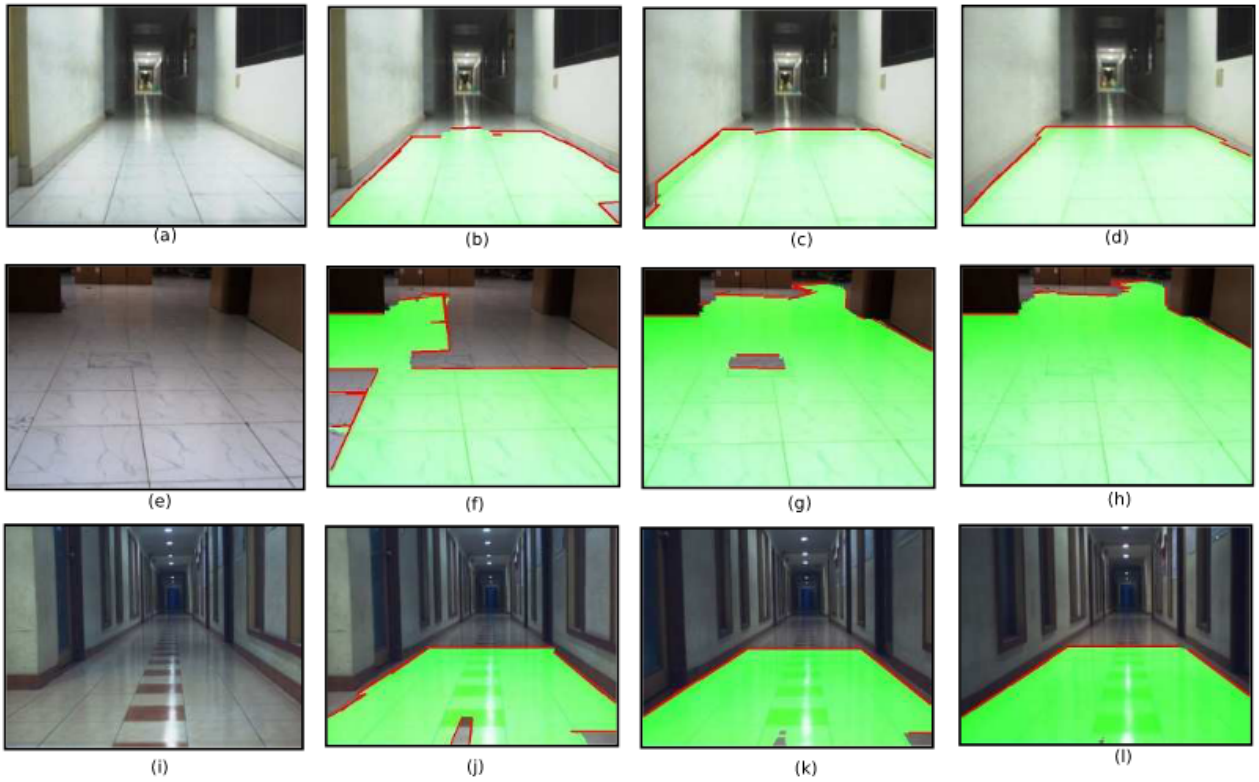


Figure 8: Detection of the ground plane improves as the probability propagates with successive iterations and pixels gain confidence. (a)-(d) are images of a wall and floor with homogeneous colors, (e)-(h) represent a room consisting of mostly floor and the resulting low confidence due to lack of feedback, (i)-(l) demonstrate the effects on a textured floor.

# References

[1] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html

[2] https://cseweb.ucsd.edu/classes/wi07/cse252a/homography_estimation/homography_estimation.pdf

[3] http://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf

[4] https://homes.cs.washington.edu/~shapiro/EE596/notes/Optical_Flow.pdf

[5] https://dl-acm-org.proxy.lib.csus.edu/doi/pdf/10.1145/2425333.2425387