Edge based approach to estimating optical flow parameters from drone images

Ajith Anil Meera
Masters student, 3mE
Delft Institute of Technology
The Netherlands
ajitham1994@gmail.com

Guido de Croon
Assistant Professor, Aerospace Engineering
Delft Institute of Technology
The Netherlands
g.c.h.e.decroon@tudelft.nl

Martijn Wisse
Professor, 3mE
Delft Institute of Technology
The Netherlands
M.Wisse@tudelft.nl

***Abstract*—The contemporary techniques employed to estimate the optical flow parameters between a pair of consecutive images utilize the entire image pixels for image correspondences using non linear optimization techniques, which makes the algorithm computationally heavy, especially for real time applications with on-board computing, like on drones for example. This paper presents an edge based algorithm to evaluate the flow parameters in a way that is sparse and computationally cheap - without having to match every pixel in the image - by evaluating those edge pixels in the image with highest confidence for a flow to have occurred along the direction of the tangent to the edge at that point. The algorithm handles the noise in the estimated flow vectors from disrupting and destabilizing the solution by iteratively removing them. The algorithm was validated by artificially generating an image database with known flow parameters and comparing the results of the algorithm to it. The results of the algorithm were found to be better than the Lukas Kanade method applied on Harris corners.**

***Index Terms*—Optical flow, computer vision, micro aerial vehicles**

I. Introduction

Optical flow is the displacement field between two images emerging as a result of an apparent motion between the observer and the object. Determination of the optical flow or the image correspondence is one of the most important step towards solving many of the real life computer vision problems including visual odometry [1], [2], SLAM [3], 3D reconstruction, structure from motion etc. It has found wide application in the autonomous flight of micro aerial vehicles where the flow parameters like Focus of Expansion (FoE), time to contact etc are evaluated and used for autonomous flight or landing of drones [4], [5]. Multiple versions of image correspondence algorithms were proposed over the years including the classic ones by Lucas and Kanade [6] and Horn and Schunck [7]. Most of these methods were based on assumptions or constraints like brightness consistency, smoothness consistency etc, based on which a non linear optimization is performed to find the optimal transformation or warp from one image to the other [8]. While the accuracy of these methods are quite good, they still remain computationally expensive so as to be directly employed in real-time on micro aerial vehicles (MAVs) relying on on-board computing. Therefore, the necessity of a sparse and computationally cheaper algorithm for the estimation of optical flow parameters is high.

II. Optical flow perpendicular to edges

Focus of expansion is a hypothetical point on the image towards which the observer is moving towards or away from the objects in scene, as seen through the camera. In such cases of camera moving towards or away from the scene, the optical flow at different points on the image would collectively intersect at the FoE.

The state of the art techniques compute the FoE by using the optical flow vectors in the entire image. In this algorithm, instead of utilizing the entire image pixels to evaluate the FoE, only the edge pixels in the image are considered. This is based on the idea that edges contain more information about the relative motion between frames than other bulky blobs of areas without much of a gradient change. The component of the optical flow at these edge pixels in the direction perpendicular to the edges are then evaluated. Even though the optical flow thus computed will only be a component of the real flow vector at these pixels, these components can collectively represent the real FoE approximately [9]. Therefore, these vector components can be used to approximately estimate the location of FoE without having to search for the real pixel matches between the pair of images, which is computationally expensive.

A. Gradient direction and edge detection

The image gradients along the x and y axes were computed using the Sobel x and Sobel y filters respectively. The gradient direction at a particular pixel in the image is calculated by using the formula,

$$\theta = \tan^{-1} \frac{dY}{dX} \tag{1}$$

where dX and dY are the image gradients along x and y directions respectively. The prominent edge pixels in the image are then evaluated by thresholding the absolute gradient along x and y direction. Those pixels with strong gradient along either x or y direction are considered as a prominent edge.

B. Search space for pixel matching

Pixel matching at all the dominant edge pixels in the image can be done accurately using computationally expensive algorithms like Lucas Kanade. However, while searching in the direction perpendicular to an edge applying these methods

might not yield the best results compared to a simple block matching algorithm. The search space of a block matching algorithm can be simplified to a square - as shown in figure 1 - centered at the edge pixel on the first image with the square's side length of $2 * search\_range$, where $search\_range$ is the number of pixels to the left and right side of the edge pixel to be searched for, for a possible match. Using a square search space considers a non-uniform distribution of points around the edge pixel where more exploration is done along the diagonal of the square while a more crowded and detailed search is done along the x and y axis. An alternate search space would be a circle around the edge pixel where the search is uniform in all directions.

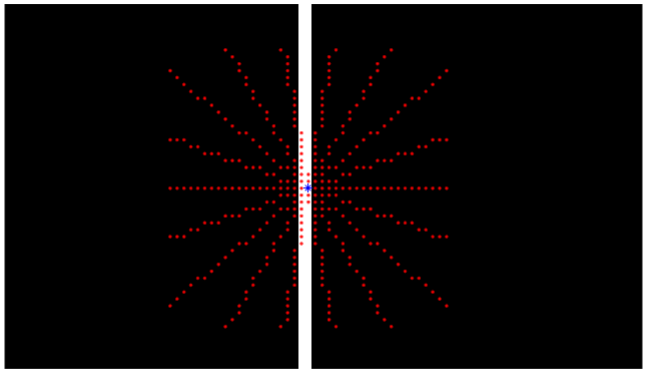

Fig. 1: The square shaped search space for pixel matching marked in red at a pixel marked in blue at one of the edge points marked by a white line in the black background

C. Block matching

Block matching was performed along the direction perpendicular to the edge in the search space using SSD (Sum of squared difference) and SAD (Sum of absolute difference) correlations. The pixel with the minimum correlation was identified as the best match using a sliding window approach. The Eucledian distance between the edge pixel and the best match is used as the magnitude of the optical flow vector at that point. It is possible that the best match is found in the direction that is 180deg out of phase with the gradient direction since the best match is searched for in both sides of the pixel. In such cases the optical flow is updated to be in the opposite direction as that of the image gradient direction.

D. Sub-pixel flow

The best matches found through block matching results in discrete flow magnitudes. This would mean that the flow is not smooth and hence would not preserve sub pixel information about the flow [10], leading to severe biases [11]. Therefore, it is important to compute sub-pixel flow.

The distribution of the SSD correlation around the global minimum can be approximated to be a quadratic one, as can be observed from figure 2. Block matching gives a result of 6 for the flow magnitude, while a quadratic fit over the immediate vicinity of the global minimum gives a result of 6.3477, which is the sub-pixel flow at that point. In this manner, a better estimate for the flow is reached by fitting a parabola to the SSD correlation.

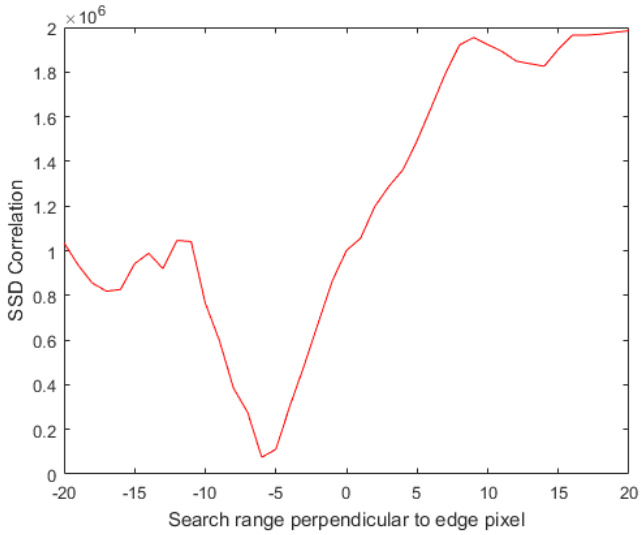

Fig. 2: Distribution of SSD correlation along the search range for the pixel [161,386] in the image shown in figure 5

E. Optical flow results

The results of the algorithm for a pair of images captured from a video frame in which the drone/camera was rolling or rotating in the clockwise direction about its axis pointed towards the front is shown in figure 3. It can be observed that the optical flow at edges effectively captures the motion of the drone.

The results of the algorithm for a pair of images captured from a video frame in which the drone/camera was moving towards the centre of the frame is shown in figure 5. It can be observed that the optical flow vectors are all perpendicular to the image edges. The flow in the left side of the image points to the right, while the flow in the right side points to the left and the bottom one to the top, which is intuitively correct because all these edges are effectively moving towards the centre of the image in a zooming out video. It can also be observed that the magnitude of flow vectors are very low or zero when the edge is parallel to the direction of flow - the edges at the rooftop at about 135deg for example has very low flow magnitude. Therefore, if the actual flow direction aligns with the perpendicular direction of the edge, the flow magnitude at those edge points is higher and if it is along the edge, the flow magnitude is lower. However, the result is not perfect and contains noises with peaks of erroneous flow vectors emerging in between.

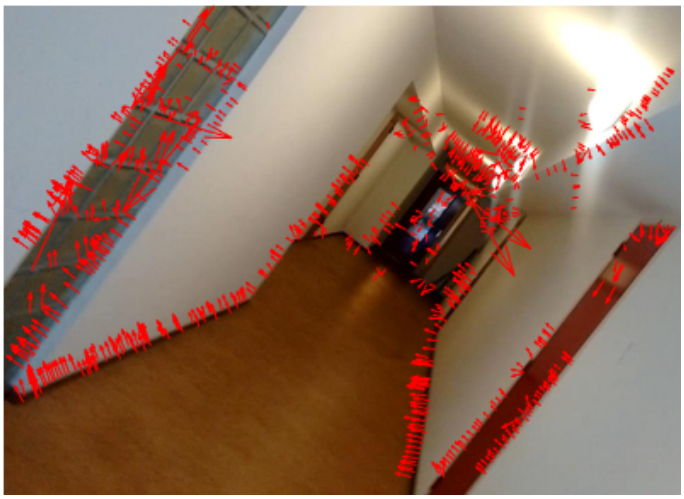

Fig. 3: Optical flow with sub-pixel flow on edge points when the drone was rolling in the clockwise sense.

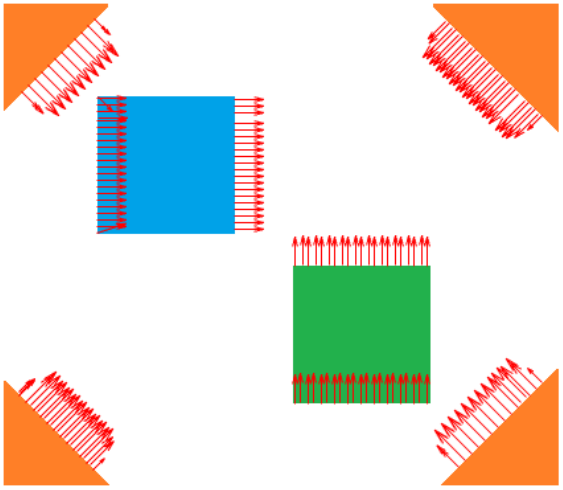

Fig. 4: Validation of the flow calculation: blue square was moved 10 pixels towards right, green square 10 pixels upwards, orange border 10 pixels each in x and y direction into the image centre. The obtained flow magnitude and direction matches with the real value. No flow at points where flow aligns with the tangent to the edge.

F. Validation of the flow

Since the computed flow is only a component of the real flow, the existing validation databases cannot be used to validate the flow. Therefore, artificial images were created and motion of edges were made exclusively in the direction normal to the edges. The optical flow was evaluated using the algorithm and was compared to the real motion to validate the algorithm. It can be observed from figure 4 that all the flow vectors are normal to the edges. The magnitude and direction of the flow thus evaluated were matching exactly with the real flow, except for the pixels that lie towards the boundary of the image. No flow was observed at those pixels where the flow is parallel to the edge - top and bottom edges of the blue square and left and right edges of the green square in figure 4. This validates the optical flow algorithm.

III. Estimation of Focus of Expansion (FoE)

In practice the FoE is estimated by taking the intersection of all flow vectors in the image. In case of a translation of the drone parallel to the scene, the FoE would be at infinity. In case of a translation purely in the forward axis of the drone, the FoE would be at the centre of the image. However, since our algorithm does not contain the actual flow directions at edge points, taking this approach would result in the centroid of all the edges in the image instead of an FoE. Therefore, a different approach is necessary to evaluate the FoE. However, it should be noted that there are methods that made use of all the flow vectors perpendicular to the edge to evaluate FoE [9].

A. Flow vectors of zero magnitude

If the actual flow is purely in the direction normal to the edge, then the computed flow direction and magnitude is the same as that of the actual flow. Similarly, if the actual flow is purely along the edge, then the computed flow magnitude would be zero. However, having a zero magnitude flow might not necessarily mean that there is a flow in the direction tangent to the edge. It could also mean that there is no flow at all in any direction at that point. However, given that the drone is moving, it is most likely that a zero magnitude flow represents a flow purely in the direction along the tangent at the edge. Making use of this principle helps in collecting all the possible candidate flow vectors from a bunch of computed flow which could actually represent the real flow at those points. The intersection of all the lines obtained by extending these flow vectors thus collected would approximately represent the FoE.

B. Least means square solution

Given the edge points with a computed flow of zero magnitude in the direction normal to the edge and the orientation of the edge at that point, it is possible to evaluate the line containing the actual flow vector that orients along the edge. Intersection of a large number of such lines would not necessarily be a point, instead it would a collection of intersection points. The best estimate of the intersection of these lines can be found by solving the least square problem of intersection of a number of constraint equations of the form,

$$y = \tan\theta_1 x + (y_1 - \tan\theta_1 x_1)$$

$$y = \tan\theta_2 x + (y_2 - \tan\theta_2 x_2)$$

....

$$y = \tan\theta_n x + (y_n - \tan\theta_n x_n)$$

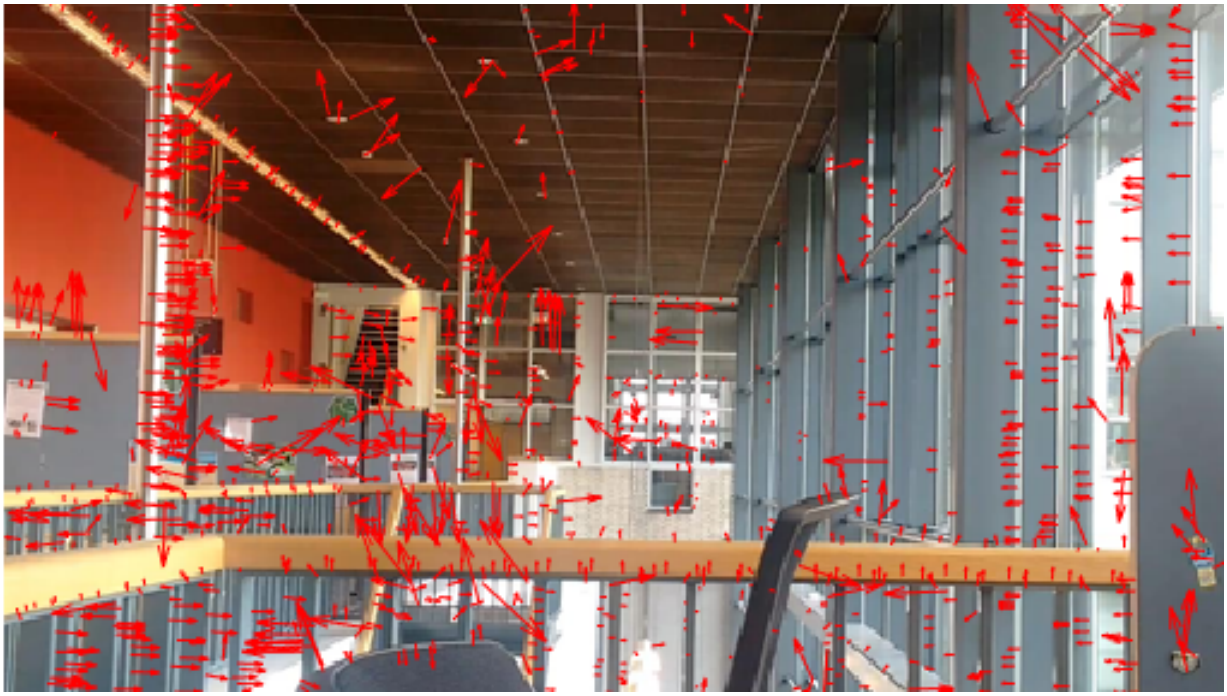

Fig. 5: Optical flow on edge points in the direction perpendicular to the edges of the image computed using SSD correlation with a window size of 15 and search range of 20. For better visualization not all flow vectors are plotted and the magnitude of flow vectors are doubled

where θ_n is the orientation of the edge at the edge point (x_n, y_n). The intersection can be found by reducing the set of equations into the form,

$$\begin{bmatrix} -\tan\theta_1 & 1 \\ -\tan\theta_2 & 1 \\ \dots & \dots \\ -\tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \begin{bmatrix} y_1 - \tan\theta_1 x_1 \\ y_2 - \tan\theta_2 x_2 \\ \dots \\ y_n - \tan\theta_n x_n \end{bmatrix} \tag{2}$$

or simply put $AX = B$ where

$$A = \begin{bmatrix} -\tan\theta_1 & 1 \\ -\tan\theta_2 & 1 \\ \dots & \dots \\ -\tan\theta_n & 1 \end{bmatrix}, B = \begin{bmatrix} y_1 - \tan\theta_1 x_1 \\ y_2 - \tan\theta_2 x_2 \\ \dots \\ y_n - \tan\theta_n x_n \end{bmatrix} \; and \; X = \begin{bmatrix} x \\ y \end{bmatrix}$$

The solution can be found by finding the least mean square solution given by,

$$X = (A^T A)^{-1} A^T B \tag{3}$$

where the resulting X is the FoE.

The result of the algorithm applied on an image pair is shown in figure 6. It can be observed that the least square estimate of the FoE was found at the intersection of a large number of lines. Due to the presence of a number of outlier lines, the FOE estimate has an error of 28.1684 pixels. The FoE error is calculated by the Eucledian distance between the real FOE and the estimated FoE.

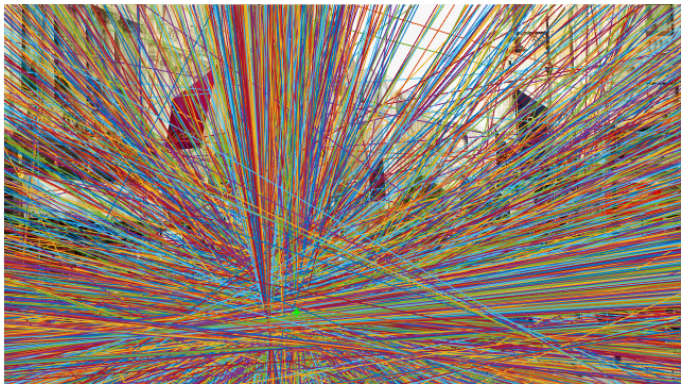

Fig. 6: Estimated FOE marked by green star. FoE error = 28.1684 pixels

C. Removal of outlier lines

The least mean square solution might give the optimal intersection of lines. However, it does not ensure an accurate solution in case a large number of outlier lines are present. Therefore, it is necessary to systematically identify and remove these outlier lines before computing the FOE. RANSAC (Random sample consensus) algorithm [12] was used in two different forms - one after the other - to remove the outlier lines. In the first version of RANSAC, the outlier lines were those lines that are far away from the FOE. This would mean

that all those lines that are farthest away from the least mean square solution would be considered as an outlier and will be removed from any further consideration. The perpendicular distance d_i from the FOE $[FOE_{xi}, FOE_{yi}]$ to a line passing through the edge pixel $[x_i, y_i]$ with an edge orientation of θ_i can be evaluated from the formula,

$$d_i = \left| \frac{\tan\theta_i FOE_{xi} - FOE_{yi} + y_i - x_i \tan\theta_i}{\sqrt{1+\tan^2\theta_i}} \right| \quad (4)$$

The model for the algorithm was the set of lines that minimizes the sum of perpendicular distances from the FOE to all of these lines given by

$$d = \sum_{i=1}^{n} d_i \quad (5)$$

All the lines that has a d_i that is atleast 2.4 times the standard deviation away from the mean of all d_i are considered as an outlier and is removed in each iteration. The loop continues till the mean of all d_i in the current iteration goes less than 3 pixels or if the loop runs out of certain number of iterations - 70 in the algorithm - whichever happens first. In this way, a large number of outlier lines that distort the accuracy of the solution are removed.

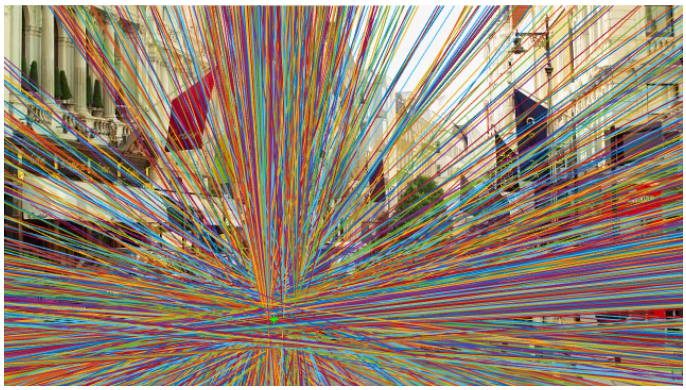

Fig. 7: Estimated FoE after performing first step RANSAC marked by green star. FoE error = 6.2818 pixels

The results of the algorithm for a pair of images is shown in figure 7. In comparison with figure 6, it can be observed that a large portion of oulier lines have been removed and that the lines are more in consensus with the intersection of the lines. The error in FoE estimate dropped down to 6.2818 pixels after performing first step RANSAC compared to 28.1684 pixels without RANSAC.

Consider three lines intersecting two at a time to form a triangle. The least square solution for the intersection of lines would approximately be within this triangle. If the triangle is too small, like in the case of figure 7, the FoE estimate would be very close to accurate. However, if the triangle is large enough, the error would be quite high. The correct solution to the problem might be the intersection of a pair of lines, instead of intersection of three lines. Moreover, the first step RANSAC will not be able to solve this problem because all these lines would be almost equidistant from the least square solution. This would mean that they are no longer an outlier for the algorithm and the solution immediately converges to an inaccurate solution. Therefore, it is important to remove such an outlier line from the solution to improve the accuracy.

A second step RANSAC which is more similar to the original version of the RANSAC algorithm can then be performed on the lines so that a fixed number of lines - 200 for the algorithm - are randomly drawn from the resulting lines of first step RANSAC and the set of lines with the least sum of distance d is used to finally estimate the FoE. This would mean that if the first step RANSAC resulted in a set of lines that intersects forming a triangle with 3 major intersections, then the second step RANSAC would sample from this set to find the most prominent intersection among all three. However, if the triangular intersection is too small, second step RANSAC might not bring any improvement in FoE estimate.

The result of the second step RANSAC is shown in figure 8. 200 sample lines were drawn randomly in each iteration for 500 iterations to find the set of lines with minimum sum of distances from the estimated solution. The error dropped further to 3.2366 pixels compared to 6.28188 pixels after first step RANSAC and 28.1684 pixels without RANSAC.

Fig. 8: Estimated FoE after performing second step RANSAC marked by green star. FoE error = 3.2366 pixels

IV. Validation of FoE estimate

A. Image database

In order to validate the algorithm a database with pairs of images with labeled FoE is necessary. Such a database was created using an image and by artificially creating its pair with a particular FoE. An artificial image with a particular FoE was created by placing the image on a new blank image such that the new image centre is at the FoE of the old image. The new image was then scaled up by a factor - 1.02 in the algorithm - representing the forward velocity of the drone. The image is then cropped to bring it to the initial image size. The location of FoE in the image represents the orientation at which the drone is approaching the scene. In this way, the

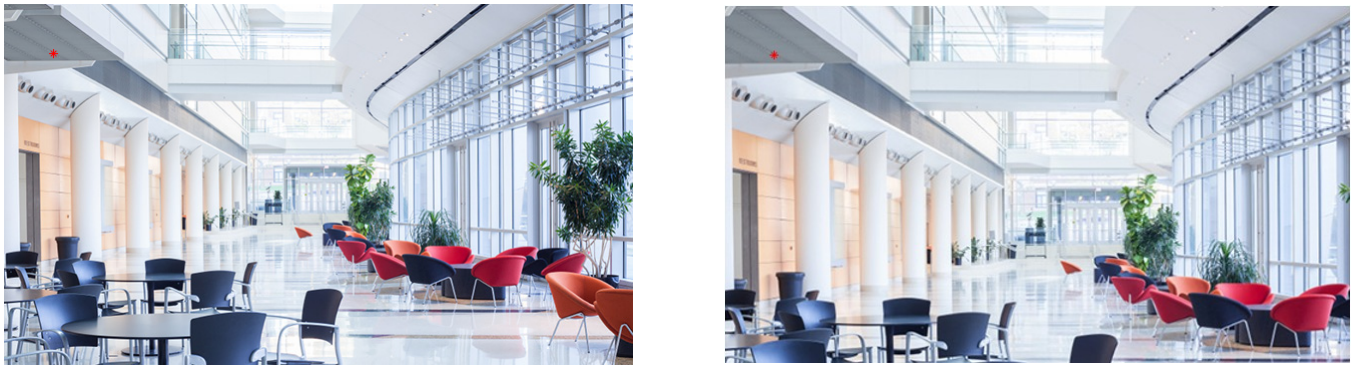

Fig. 9: Artificial image generated from the first image with scaling factor of 1.2 and FoE at [50,50], marked by a red star.

artificial image is created by considering the forward velocity and heading orientation of the drone.

The result of the generation of validation dataset from an image for an FoE of [50,50] with a scaling factor of 1.2 is shown in figure 9. It can be observed that the image is scaled up with respect to the pixel [50,50] due to which the plant in the right side of the first image is missing from the second image. Moreover, the features in the left side of the image are still inside the image except for the bottom portion of the first image.

B. Heat map of FoE estimate error

Validation of the algorithm should be performed over a number of locations of FoE on the same image to observe if the algorithm behaved the same throughout the image. Therefore, a heat map of the FoE estimate error was generated by moving the FoE location throughout an image with a particular stride and then generating the corresponding artificial image upon which the algorithm is tested and the error is reported.

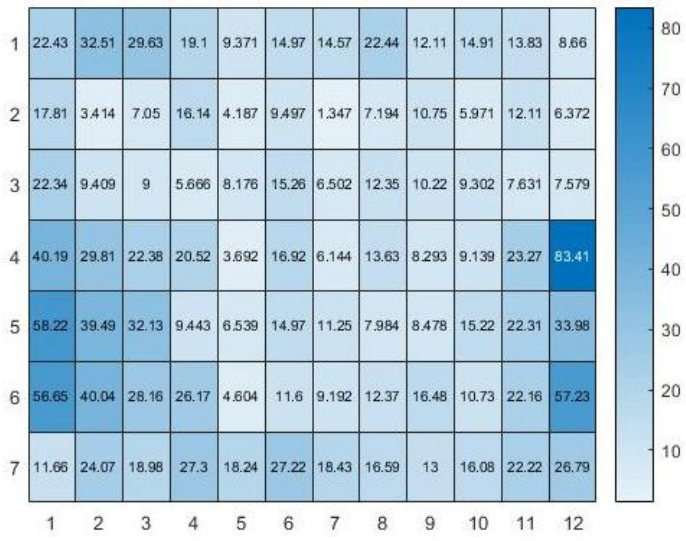

Fig. 10: Heat map of FoE estimate error at different FoE for the street image in figure 8. Mean error = 17.720 pixels

Figure 10 shows the heat map of the street image in figure 8 with a stride of 50 pixels in both x and y direction. This means that the error in the heat map corresponding to location (3,7) represents the FoE error estimate when the FoE was kept at the location [150,350] in the street image. It can be observed that the error is not the same throughout the image which is intuitive because the algorithm is dependent on edges and the edges in the image are not uniform - edges are oriented in different directions. Sudden increase in the error at particular locations in the image can also be observed, FoE at [200,600] yields an error of 82.41 pixel for example. This is a location where the outlier lines were too high that they still remained in the algorithm. Tuning the parameters of RANSAC algorithm can fix this error easily. However, the FoE error at most of the other locations remained satisfactorily low.

The importance of RANSAC algorithm can be visualized from the error heat map generated without using RANSAC shown in figure 11 and comparing it to figure 10.

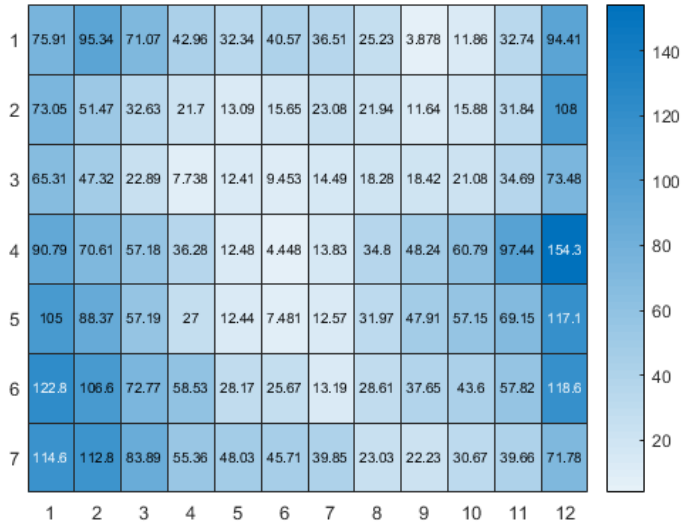

Fig. 11: Heat map of the FoE estimation error at different locations of FoE for the street image in figure 8. Mean error = 48.149 pixels

It can be observed that the error is much higher when RANSAC is not used to evaluate the FoE for almost all elements in the heat map. Moreover, the mean error was reduced from 48.149 pixels to 17.720 by using the RANSAC algorithm.

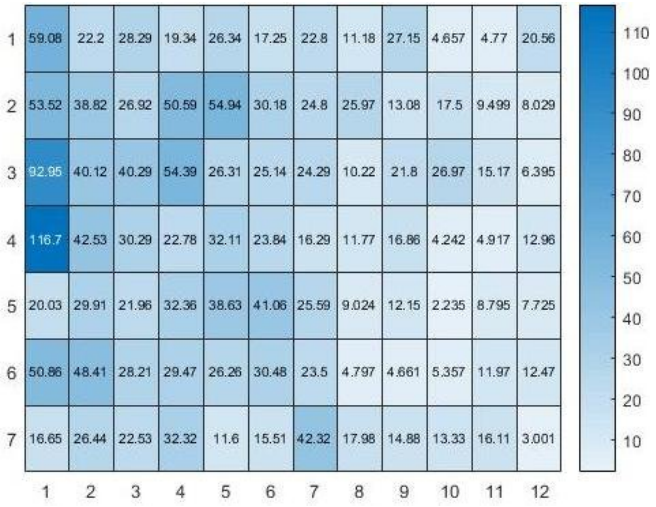

Fig. 12: Heat map of the FoE estimation error for the indoor image in figure 9.

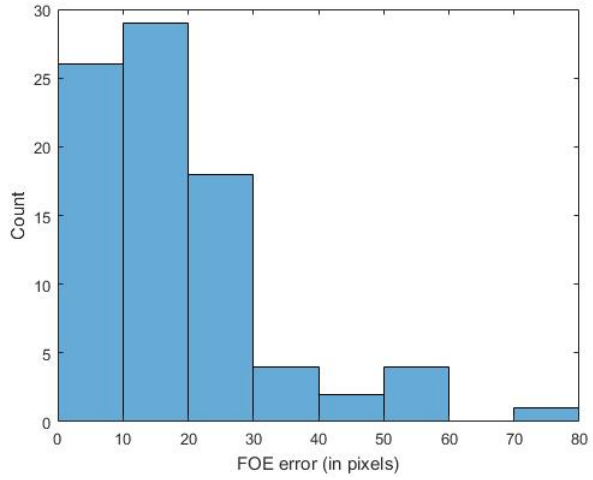

Fig. 13: Histogram of heat map shown in figure 10 for the street image. Bin width = 10 pixels.

C. Histogram of FoE estimate error

In order to analyze the spread of FoE estimate error, the histogram of the estimated error was plotted from the heat map. High number of points in the first bin and lower number of points on the higher bins represent a good algorithm for FoE estimation. Fewer number of points on the higher bins would indicate the robustness of the algorithm. The histogram of the FoE error for street image is shown in figure 13. It can be observed that most of the points belong to the first two bins and only few points belong to the highest bins.

D. Comparison to Lukas Kanade method

In order to compare the results of the algorithm with that of Lukas Kanade (LK) method, the strong corners in the image were selected using Harris Corner detector and the real optical flow was evaluated at these points using Lukas Kanade method. Since these flow vectors represent the real flow, they can directly be used to find the FoE. The intersection of lines passing though the real flow vectors represent the FoE. For the sake of comparison the number of corner points used were taken to be nearly the same as that of the number of flow vectors with zero flow magnitude in the normal direction to the edge, that was used for the algorithm. This would mean that the number of lines used to evaluate the FoE for our algorithm and for LK method are the same. Moreover, RANSAC was applied on the lines obtained through LK method, using exactly the same parameters and code. The heat map and the histogram for the street image are shown in figure 14 and 15. It can be observed that the errors are much higher than that of our algorithm shown in figure 10. The higher bins of the histogram shown in figure 15 have a very large error compared to the histogram generated through our algorithm as shown in 13. The major share of the error occurs at the corners of the heat map, which resulted in a high mean error of 102.156 pixels. Similar trend can be observed in the histogram of the indoor image shown in figure 16 that uses LK method.

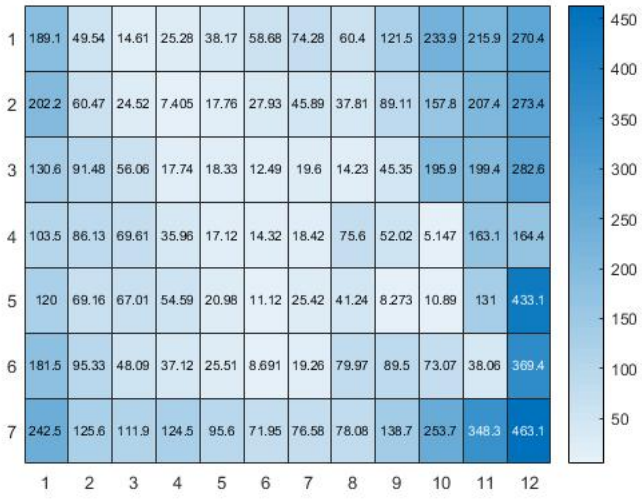

Fig. 14: Heat map of the FoE estimation error using Lukas Kanade method at the Harris corners on the street image in figure 8. Mean error = 102.156 pixels

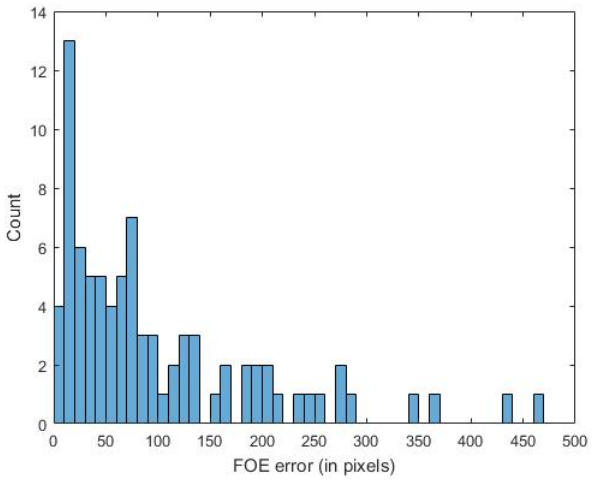

Fig. 15: Histogram of heat map shown in figure 14 for the street image. Bin width = 10 pixels

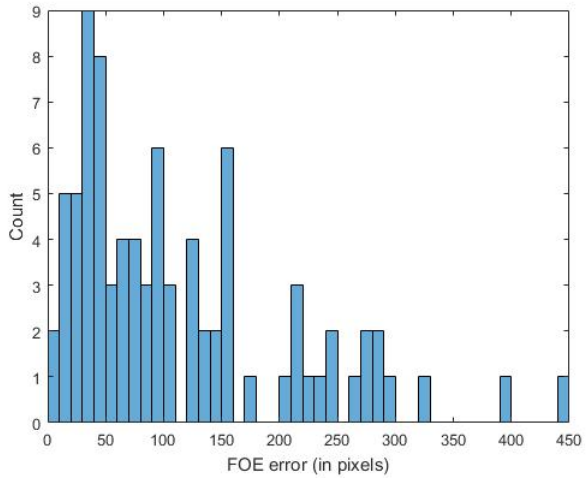

Fig. 16: Histogram of heat map for the indoor image shown in figure 9. Bin width = 10 pixels

V. Conclusion

An edge based algorithm to evaluate the optical flow parameters of the drone images was introduced. Instead of matching all the pixels in the image pairs, the algorithms matches only the prominent edge pixels thus reducing the computation time for the optical flow estimation. This restricted information about the flow only in the direction normal to the edges were enough to estimate the flow parameters with satisfactory accuracy. The algorithm is robust to outliers and noise in the data which was achieved by systematically removing the outliers using a two step RANSAC algorithm. The algorithm was validated by running it over a number of images for FoE located throughout the image and reporting the heat map and histogram for FoE estimate error. Although there were some erroneous results, on an average the results were good enough to be run on a drone for tasks like obstacle avoidance. The importance of RANSAC algorithm was demonstrated by comparing the error heat map with and without using RANSAC. The results of the algorithm were compared to the LK method applied on Harris corners. Our algorithm demonstrated a better performance in terms of estimated FoE error.

However, the accuracy of the algorithm still needs to be improved. Moreover, the validation can be performed on real video stream from the drone with true FoE information that can be computed by tracking the position of the drone through accurate tracking systems. This would also test the algorithm on handling the distortion effects of the camera, which were not considered while generating the artificial image for the validation database.

VI. Code for the algorithm

The codes for the entire algorithm is available in two languages - MATLAB and C - freely at
$https://github.com/ajitham123/Optical\_flow$.
The C code is available inside the $c\_optical\_flow$ folder and is named $main.cpp$. Library requirement for the C code includes the OpenCV library used for image processing. The code for image database generation was written only for MATLAB. The link also contains all the images and videos used for testing and validating the algorithm. In case of any confusion in accessing the codes please write to $ajitham1994@gmail.com$

References

[1] Engel, Jakob, Jurgen Sturm, and Daniel Cremers. "Semi-dense visual odometry for a monocular camera." Proceedings of the IEEE international conference on computer vision. 2013.

[2] Kerl, Christian, Jrgen Sturm, and Daniel Cremers. "Robust odometry estimation for RGB-D cameras." Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013.

[3] Engel, Jakob, Thomas Schps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." European Conference on Computer Vision. Springer, Cham, 2014.

[4] Izzo, Dario, and Guido De Croon. "Landing with time-to-contact and ventral optic flow estimates." Journal of Guidance, Control and Dynamics 35.4 (2012): 1362-1367.

[5] De Croon, G. C. H. E., et al. "Optic-flow based slope estimation for autonomous landing." International Journal of Micro Air Vehicles 5.4 (2013): 287-297.

[6] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674-679.

[7] Horn, Berthold KP, and Brian G. Schunck. "Determining optical flow." Artificial intelligence 17.1-3 (1981): 185-203.

[8] Brox, Thomas, et al. "High accuracy optical flow estimation based on a theory for warping." Computer Vision-ECCV 2004 (2004): 25-36.

[9] Hordijk, Bas J. Pijnacker, Kirk YW Scheper, and Guido GCE de Croon. "Vertical Landing for Micro Air Vehicles using Event-Based Optical Flow." arXiv preprint arXiv:1702.00061 (2017).

[10] Szeliski R., Scharstein D. (2002) Symmetric Sub-pixel Stereo Matching. In: Heyden A., Sparr G., Nielsen M., Johansen P. (eds) Computer Vision ECCV 2002. ECCV 2002. Lecture Notes in Computer Science, vol 2351. Springer, Berlin, Heidelberg.

[11] Shimizu, Masao, and Masatoshi Okutomi. "Precise subpixel estimation on areabased matching." systems and computers in Japan 33.7 (2002): 1-10.

[12] Fischler, Martin A., and Robert C. Bolles. ”Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.” Communications of the ACM 24.6 (1981): 381-395.