N94- 35055

Accurate Estimation of Object Location in an Image Sequence Using Helicopter Flight Data[1]

Yuan-Liang Tang and **Rangachar Kasturi**[2]

Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
Phone: (814) 863-4254
Email: *kasturi@cse.psu.edu*

Abstract

In autonomous navigation, it is essential to obtain a three-dimensional (3D) description of the static environment in which the vehicle is traveling. For rotorcrafts conducting low-altitude flight, this description is particularly useful for obstacle detection and avoidance. In this paper, we address the problem of 3D position estimation for static objects from a monocular sequence of images captured from a low-altitude flying helicopter. Since the environment is static, it is well known that the optical flow in the image will produce a radiating pattern from the focus of expansion. We propose a motion analysis system which utilizes the epipolar constraint to accurately estimate 3D positions of scene objects in a real world image sequence taken from a low-altitude flying helicopter. Results show that this approach gives good estimates of object positions near the rotorcraft's intended flightpath.

1 Introduction

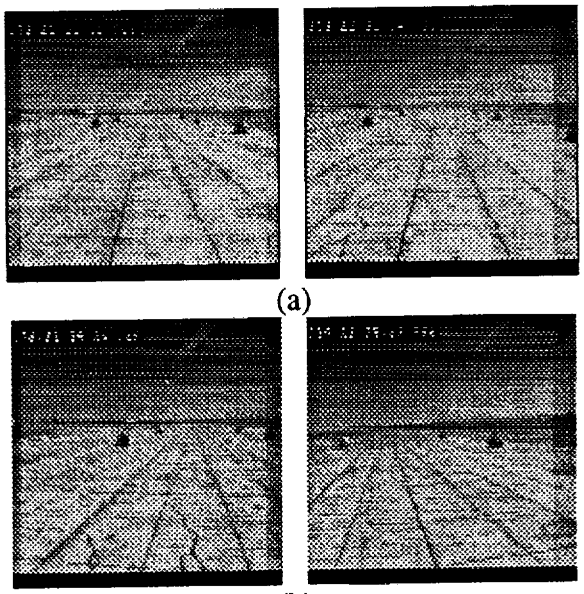
To relieve the heavy workload imposed upon the pilots, there is a need for automatic obstacle detection systems onboard rotorcrafts. The success of the system depends upon the ability to accurately estimate object positions near the rotorcraft's flightpath. Several approaches for obstacle detection and range estimation have been investigated at NASA Ames Research Center [Bhanu89, Cheng91, Roberts91, Smith92, Sridhar89]. In this paper, we propose an approach for object position estimation using known camera location and motion parameters.

For a rotorcraft with inertial-guidance systems, the information about camera state is continuously available as the rotorcraft moves. This information can thus be used to facilitate the processes of motion estimation and scene reconstruction. For example, the location of the focus of expansion (FOE) in the image plane can be readily determined. In addition, we assume the image acquisition rate is high enough that an image feature will not move by more than a few pixels in the next image frame. Such closely sampled images will minimize the correspondence problem between successive images. The forward moving camera situation is the worst case in depth estimation because the optical flow in the image is small compared to other motion cases. We overcome this

[1]This research is supported by a grant from NASA Langley Research Center, Hampton, Virginia (NAG-1-1371).
[2]Address all correspondence to Professor Kasturi.

problem by integrating information over a long sequence of images. As image frames are accumulated, the base line between the current frame and the first frame increases, which gives better motion estimates. Baker and Bolles [Baker89, Bolles87] used the *Epipolar Plane Image (EPI) Analysis* for motion analysis. In their approach, camera moving path is known and linear. Therefore, each image frame can be decomposed into a set of epipolar lines. An epipolar plane image can thus be created by collecting corresponding epipolar lines in each image frame. Furthermore, when the viewing direction is orthogonal to the direction of motion, the apparent motion track of a feature on the EPI is a straight line and the motion analysis becomes merely a line fitting process. For forward linear camera motion, however, the feature tracks will be hyperbolas and curve fitting becomes necessary. Sawhney et. al. [Sawhney93] have reported that curve fitting is much more difficult and noisy, making this approach less robust. Matthies et. al. [Matthies89] built a framework which gives depth estimates for every pixel in the image. Kalman filtering is used to incrementally refine the estimates. In their experiments also, the side-viewing camera is assumed and the camera motion is only translational in the vertical direction. Under such conditions, feature tracks will follow the vertical image scan lines and feature matching becomes simpler. In our situation, the problem of general camera motion is dealt with. We handle this by breaking the camera motion path into piece-wise linear segments. Through this process, the camera path determined by two consecutive camera positions is approximated as a straight line. Epipolar planes can thus be set up for each pair of images and motion analysis is recursively performed on each pair of image frames.

Our algorithms were tested on the helicopter images provided by NASA Ames Research Center [Smith90]. There are two sequences of images, namely the *line* and the *arc* sequences. Each sequence consists of 90 image frames with size 512x512 pixels and each frame contains a header information which records the helicopter body and camera positions and orientations, body and camera motion parameters, camera parameters, etc. Time stamps are projected directly on the image frames. Fig. 1(a) and 1(b) show the first and the last frames of the *line* and the *arc* sequences, respectively. For the *line* sequence, the helicopter's flightpath is approximately a straight line and there are five trucks in the scene during the whole sequence. For the *arc* sequence, the helicopter is making turning flight and truck 1 is not visible in all frames. The trucks are labeled in terms of their range (X) value; truck 1 is the nearest and truck 5 is the farthest. Ground truths for the positions of the trucks are also given.


(a)

(b)

Fig. 1 The first (left) and the last (right) frames of the *line* sequence (a), and the *arc* sequence (b).

The remaining of this paper is organized as follows. In Section 2, we describe how to construct the epipolar lines. Section 3 discusses the feature extraction and tracking processes. In Section 4, we present the three-dimensional position estimation by tracking outputs. Experimental results and discussions are also given. Section 5 gives the conclusion.

2 Constructing the Epipolar Lines

The epipolar constraint gives a strong tool in confining the apparent motion directions of image features. In fact, the constrained directions are determined by epipolar planes as shown in Fig. 2(a), where all the image frames share a common set of epipolar planes. Since we are dealing with general 3D camera motion, the moving path of the camera is not a straight line and its orientation is not constant during motion. Fig. 2(b) illustrates such a case, where the camera's path is an arc. Even if the camera is fixed on the helicopter, its orientation is still changing because the orientation of the helicopter body is changing during nonlinear motion. The location of the FOE also changes significantly. In this case, there is no common set of epipolar planes for all the image frames. We solve this problem by using the piece-wise linear approximation for the camera path. Between two consecutive image frames, the camera path is approximated as linear and hence a pencil of epipolar planes can be created in the 3D space which all intersect at this segment of camera path. For each pair of image frames, we first compute the camera path parameters from the input camera state data (positions and orientations). We then define a pencil of Q epipolar planes, P_i, $i=0, .., Q-1$, which all intersect at the camera path. Q determines the resolution of the 3D space and hence the number of features to be detected in the image. In our experiments, Q is set to 100. The angle between two adjacent epipolar planes P_i and P_{i+1} is equal to π/Q. The result of this process is the construction of a pencil of epipolar planes equally spaced in terms of angular orientations and they all intersect at the camera path. After creating the epipolar planes, epipolar lines on each image plane can thus be determined by intersecting the image plane and the epipolar planes. The process of creating the epipolar lines is recursively performed on each pair of image frames in the sequence. Fig. 3 shows the epipolar lines superimposed on the edge detected images. The intersection of all the epipolar lines shows the FOE. Even for the *line* sequence in which the FOE is expected to remain at the same pixel location in the image, it changes by about 25 pixels both horizontally and vertically. For the *arc* sequence, the FOE location varies by about 70 pixels horizontally and 30 pixels vertically.

3 Feature Detection and Tracking

The purpose of constructing epipolar lines are two folds: to detect features and to facilitate feature tracking. Features in an image are defined to be the intersecting points between the epipolar lines and the edge pixels detected by the Canny's edge detector [Canny86]. The features are extracted from the first image by tracing the pencil of epipolar lines, l_i, $i=0, .., Q-1$. The result will be a number of Q sets of feature points and feature detection and tracking on the following frames are completely independent among these sets. To obtain good localization of detected features, the edges in the image should be nearly perpendicular to the epipolar lines.

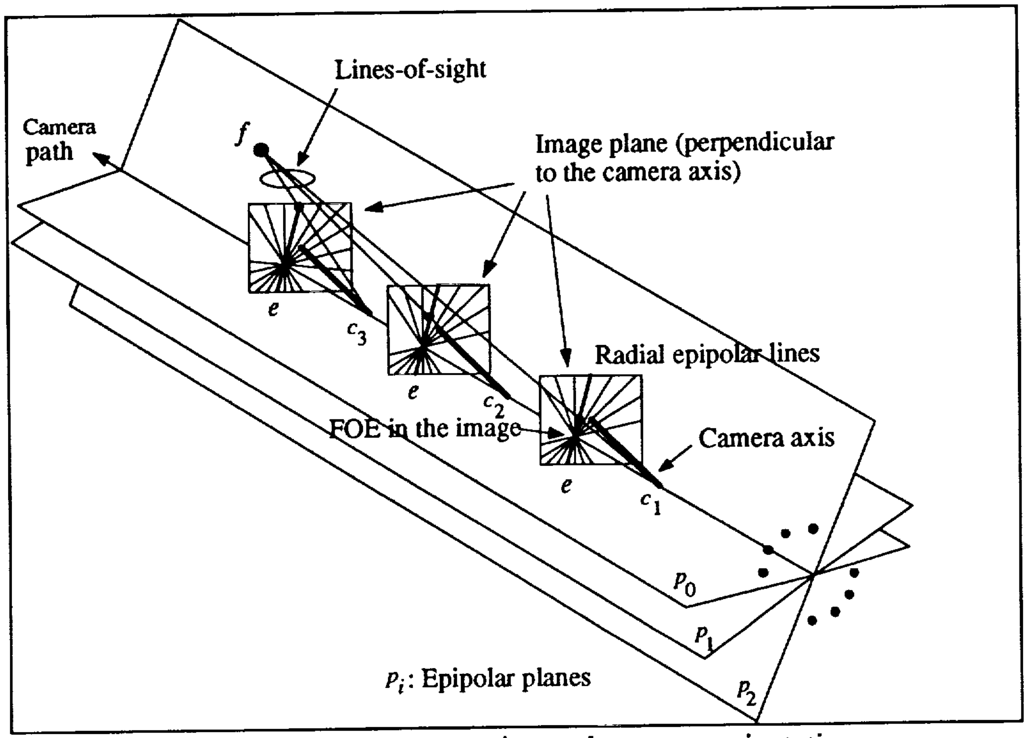


Fig. 2 Linear camera motion and constant orientation.

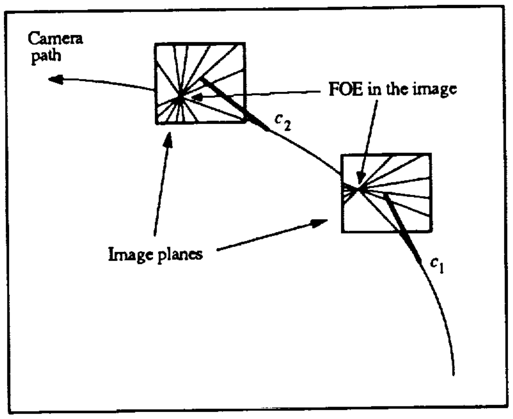


Fig. 3 Non-linear camera motion and varying orientation.

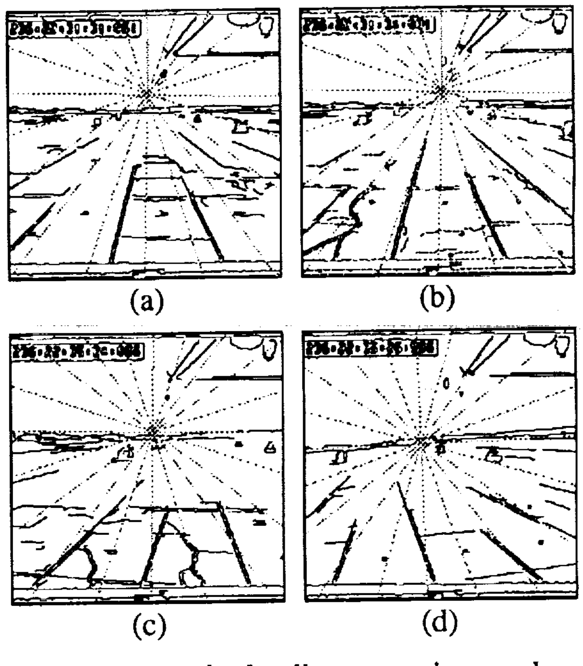


Fig. 3 The epipolar lines superimposed on the edge detected images (Every tenth line is shown). (a), (b): the first and the last frame of the *line* sequence, respectively. (c), (d): those of the *arc* sequence.

Since we are dealing with general camera motion, the camera orientations for two frames are likely to be different. Traditional feature matchers try to search within the *neighborhood*, i.e. the search window, of the feature to be matched (the source feature). With the information of camera state, we argue that this blindly positioning of the search window is inappropriate because the term *neighborhood* is incorrectly defined. The following statement is one of the implicit assumptions in defining the *neighborhood* to be a window centered around the source feature: "Since the image acquisition rate is high enough such that the camera will not move for a long distance between two consecutive frames, the source feature will not move more than a few pixels in the next image plane." We find this statement sustains only if the camera orientation keeps approximately constant during the camera motion. As is well known, even a small amount of camera rotation can actually create large image feature motion in the image plane. Hence, the camera rotation has much more influence on image feature motion compared to camera translation, especially when the distance from the camera to a world feature is very large compared to the distance the camera travels between two consecutive frames. This can be illustrated with the 2D world shown in Fig. 4, where the camera moves from c_1 to c_2 with orientation changes. The projections of a far away world feature onto the two frames are p_1 and p_2, respectively, and p_1' specifies the same image location as p_1 in the second frame. Due to camera rotation, p_2 is not, however, at the neighborhood of p_1'. Under such condition, the search window should be positioned around p_2 instead of p_1'. This is exactly our situation where the velocity of the camera is about 1 foot/frame and the major features of interest in the image are hundreds of feet away. Also, for the *arc* sequence, the instantaneous orientation of the camera is continuously changing since the flightpath is not linear, as described in Section 2. Experiments showed that if we perform tracking on the *arc* sequence by searching the neighborhood of the source feature, the correct corresponding feature may be completely out of the search window.

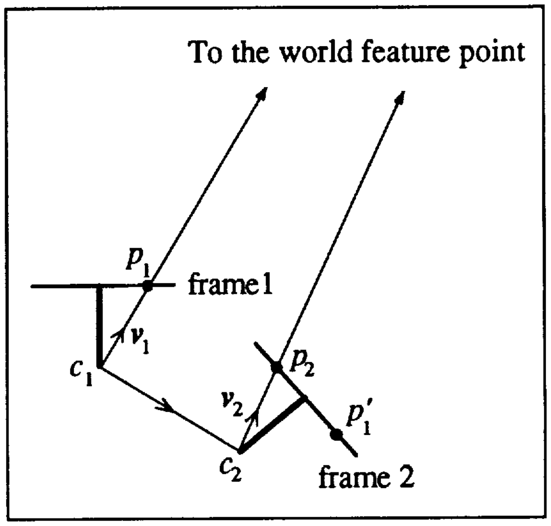


Fig. 4 Camera motion in a 2D world.

From the argument above, we redefine *neighborhood* as follows. We observe that the only thing guaranteed by high image acquisition rate is that the vector of the line-of-sight to a far away world feature will not change drastically between two consecutive frames, i.e. the vectors v_1 and v_2 in Fig. 4 should be approximately the same. Hence, the *neighborhood* of feature p_1 is redefined as "the region surrounding the intersection between the image plane of frame 2 and the vector through c_2 and parallel to v_1." And the feature tracker will search within this *neighborhood* for a match for feature p_1. This is the main reason of the success of our feature tracker. We implement the feature tracker as follows. For each pair of image frames, the locations of the FOE on each image plane are first computed. Let subscripts

1 and 2 denote the time instance of the first and the second frame, respectively. For each image feature at location p_1 in the first frame I_1, we first compute v_1 which is the 3D vector from the camera center c_1 to p_1. And then, the hypothetical location p_2 is obtained by intersecting I_2 with the vector v_1 passing through camera center c_2. Incorporating the epipolar constraint, instead of searching within a region centered around p_2, the feature tracker follows the direction of the epipolar line, which is determined by the FOE and p_2. In our experiment, we use seven pixels as the 1D window size. Feature detection and tracking are then performed within this window on the second image. Note that, in order to reduce the amount of computation we perform the feature detection and tracking based on the edge pixels only. For more robust feature tracking, the intensity distribution around a feature should be considered.

Within a small 1D *neighborhood*, we have a number of features (source features) in the first image and a number of features (target features) in the second image to be matched. Depending on the matching results, a feature will be labeled as **matched**, **new**, **no match**, and **multiple matches**:

1. **Matched**: There is only one target feature in the neighborhood. If there are several source features which are competing for the target (i.e. the occlusion case), the target will be matched to the source feature which has a stable position estimate. The position estimate of a feature is considered *stable* if its estimated position remains approximately the same through several frames (see Section 4). If more than one source feature have a stable position estimate, local slope of their tracks is compared. Fig. 5(a) shows such a situation. Source features A and B can both match to target C. The track with steeper slope (B) should correspond to the feature which is more distant from the camera than A. Since only the near feature can occlude the farther one, target C will be matched to feature A. For the matched source feature, its image (2D) position will be updated and its 3D position estimated (described in Section 4). Feature B will be labeled **no match** and handled as described later.
2. **New**: When a new feature appears in the current image, it has no source feature to match. A feature node will be created and inserted in the database.
3. **No match**: This may be due to the failure of feature detector, occlusion, or feature moving out of the image. For the last case, which can be easily detected, the source feature is simply removed. For the other two cases, if the source feature already has a stable 3D position estimate, the feature tracker will make a hypothesis about its image position based on its 3D position estimate. No estimation will be performed on these features except for updating their 2D position using the hypothesis. For other features, since we have no reliable information about their position, they remain in their current state awaiting possible matches in the future. A maximum number of consecutive no matches is defined to remove those features being occluded or missed by the detector for a long time.
4. **Multiple matches**: In this case, more than one target feature appears within the neighborhood. This may result from feature disocclusion or due to the noise from the feature detector. The goal here is to choose the best match. Cox [Cox93] reviewed some of the approaches including *nearest-neighbor* [Crowley88], *Mahalanobis* distance [Therrien89], *track-splitting filter* [Smith75], *joint-likelihood*

[Morefield77], etc. In our problem, since the epipolar constraint already gives an effective means to improve the tracking process, simple techniques are used to resolve this confusion and at the same time reduce the algorithm complexity. This idea is also supported by three observations. First, if the feature already has a stable 3D position estimate, the best match can be easily found by projecting its position estimate onto the current image. Second, according to the epipolar constraint, actually only *one* direction (away from FOE) is possible for the feature motion under noise-free circumstances. Hence, the feature motion conforming this constraint should be favored. And finally, the size of the search window is small, giving only a small number of multiple matches. Hence, the matching problem is simplified. With these observations, we use the following three priority criteria for choosing a feature to be the best match: (1) the one which satisfies the position estimate; (2) the one which is in the direction away from the FOE and is nearest; and (3) the one which is in the direction towards the FOE and is nearest. The main reason to include (3) as a legal match is to compensate the feature detection noise. These three criteria will also determine the weights in the position estimation (see Section 4). Fig. 5(b) gives a demonstration of how complicated the multiple match can be. Source feature *A* in frame 2 is searching for a best match among the target features in frame 3. Correct track is *AEFG*, but feature *E* (the blank circle) is missed by the detector. There exist also disocclusions (squares and triangles) and noise (star). Several scenarios and consequences are possible in our tracking process: (1) The tracker chooses feature *B* (the triangle) as the best match, feature *A* is already stable, and feature *F* is detected and within *B*'s search window in frame 4. Since *B* does not satisfy the estimated location of *A*, it will be lightly weighted in the position estimation and has little contribution to the estimate. However, according to the estimated position, the tracker will pick the correct match, feature *F*, as the best match in frame 4. (2) The tracker chooses feature *B* as the best match, feature *A* is stable, and feature *F* is not in *B*'s search window or is missed again. The tracker will follow the track *ABC*, which is wrong for feature *A*. The tracker, however, may still make correction on its tracking if it is possible to match feature *G* to feature *C* in frame 5. Otherwise it will follow the path of the triangles and the position estimates will gradually become unstable. Such features can be recognized by noting that their position estimates are still unstable after lengthy tracking. We then reset their estimates and start a new estimate similar to that for newly appeared features. (3) The tracker chooses *B* as the best match and feature *A* has no stable estimates. The tracking will either follow the circles or the triangles depending on which one is nearer. This does not matter too much since no reliable information has been accumulated and these errors will be lightly weighted in the position estimation. (4) The tracker chooses the noise (star) as the best match. This will be similar to what has been discussed, i.e. the tracker may correct its tracking if it is possible to pick feature *F* in frame 4. Otherwise, the estimate will be reset if track *BCD* is followed.

Feature matching is difficult and may contribute to most of the error in image analysis. From the analysis above, we can see that the epipolar constraint helps to simplify and to improve the matching quality. The matcher may be further improved by incorporating more clues such as the intensity distribution, edge strength, edge orientation, etc.

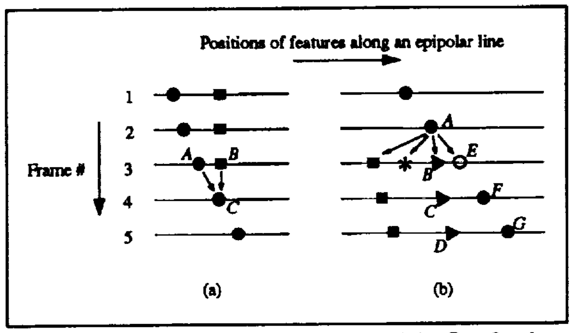


Fig. 5 Two matching cases: (a) Occlusion; (b) Disocclusion, missed features, and noise.

4 Position Estimation and Experimental Results

We use *incremental weighted least squares* for estimating the 3D position of the features being tracked. The main reason for that is its simplicity. Kalman filtering is another choice as described in [Matthies89]. However, we would like to demonstrate that using our feature tracker, a simple estimation method can still achieve accurate estimates. As we know, the line-of-sight determined by the camera center and the image feature position will also pass through the 3D position of the world feature. As time elapses, we can obtain for a certain feature a set of lines-of-sight which, theoretically, will intersect at its true 3D position. However, due to finite pixel size and inaccuracies in the known values of camera parameters and feature's image plane estimate, these do not exactly intersect at a point. This becomes a *point fitting* process. Let (a_k, b_k, c_k) be the normalized direction vector of the line-of-sight at time instance k and (X_{ck}, Y_{ck}, Z_{ck}) be the position of the camera center. The distance d_k between the 3D feature position (X, Y, Z) and the line-of-sight can be computed as:

$$d_k = \left\{[c_kY_k - b_kZ_k]^2 + [a_kZ_k - c_kX_k]^2 + [b_kX_k - a_kY_k]^2\right\}^{1/2},$$

where

$$(X_k, Y_k, Z_k) = (X - X_{ck}, Y - Y_{ck}, Z - Z_{ck}).$$

The objective function J can be defined as the sum of the weighted squares of the distances, i.e.,

$$J = \sum_{k=1}^{t} w_k d_k^2,$$

where w_k is the weight and t is the current time. J is to be minimized to obtain a least squares estimate. The weight will be determined by the result of matching. It is defined as:

$$w_k = \begin{cases} w_p, & \text{if sastifying the position estimation, else} \\ w_e, & \text{if satisfying the epipolar constraint, else} \\ w_n, & \text{if nearest,} \end{cases}$$

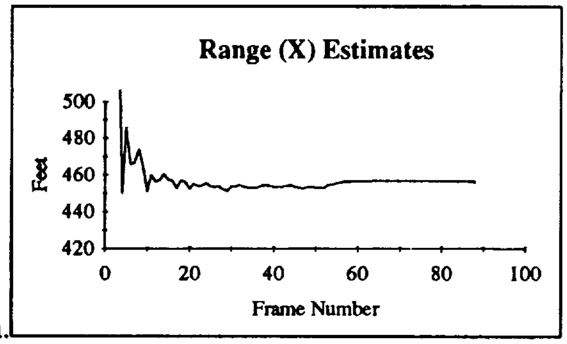
where $1 \geq w_p \geq w_e \geq w_n \geq 0$. The position estimate is considered stable at time t if

$$\frac{1}{r}\sum_{j=t-r}^{t-1}\left\{(\hat{X}_j - \hat{X}_t)^2 + (\hat{Y}_j - \hat{Y}_t)^2 + (\hat{Z}_j - \hat{Z}_t)^2\right\} < T,$$

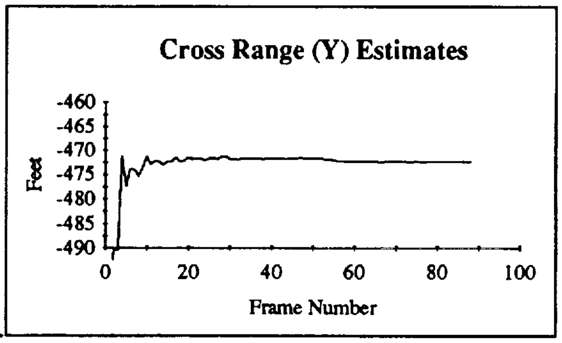
where $(\hat{X}_j, \hat{Y}_j, \hat{Z}_j)$ is the position estimate at time j, T is a threshold and r is the number of frames considered.

For the *line* sequence, the tracking for the five trucks are all successful. Fig. 6 shows the position estimates and the relative estimate errors of truck 1 (right most) as a function of number of frames processed. The relative error is defined as the difference between the estimated value and the ground truth divided by the Euclidean distance between the camera center and the ground truth. We can see from the figures that the estimation converges after 10 frames and the relative errors of the estimates for the range, cross range, and height after convergence are within 5%, 1%, and 1%, respectively. As the frame acquisition rate is 30 frames/sec, the estimate converges in less than one second. The reason for this fast convergence is that truck 1 is the nearest and the average number of pixels moved between two consecutive frames is about 0.7. The fast moving image features provide better optical flow information for motion estimation. The more accurate are the cross range (*Y*) and the height (*Z*) estimates since there is little lateral or vertical motion. While the least accurate is the range (*X*) estimate since the helicopter is moving forward. Due to this fact, in all our experiments the cross range and the height estimates are always more accurate than the range estimates and their relative errors are about 1%. Table I summarizes only the range estimates for all the five trucks in the *line* sequence. It is also shown that the number of frames needed for range estimate to converge within 10% error increases as the distance to the truck increases. Truck 5 needs more than 90 frames to obtain a more accurate estimate because it is farthest and the average number of pixels of image feature movement is only 0.1 pixel.

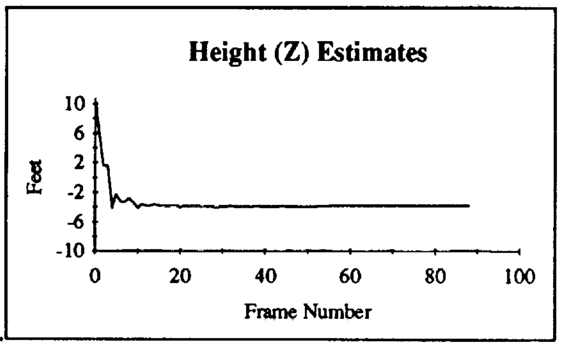
Experiments are also performed on the *arc* sequence and Table II summarizes the estimates. The tracking of truck 4 is not quite successful. The reasons being that during the initial several frames the detected edge is approximately parallel to one of the epipolar lines, making the tracking more difficult. However, its position estimate converges to within 25% errors after tracking through 90 frames. The estimate would converge to the correct value provided that more image frames are accumulated. Comparing Tables I and II, we find that the range estimates obtained from processing the *arc* sequence tend to be better than those from the *line* sequence. This verifies our previous statement that the forward moving camera case presents more difficult problem in range estimation. Since the helicopter is making a turning flight in the *arc* sequence, features move faster in the image than those in the *line* sequence, and hence provide larger optical flow for motion estimation.



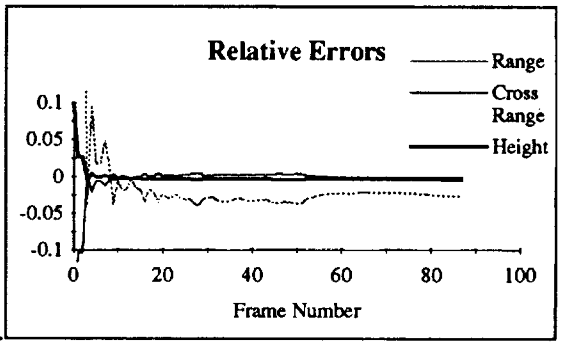
a.



b.



c.



d.

Fig. 6 Position estimates (a), (b), (c) of truck 1 and relative errors (d) as a function of the number of frames processed. The ground truth is $(X, Y, Z)=(461.5, -472.3, -3.0)$ with accuracy ±2 feet.

Table I Position estimates of various trucks after processing 90 *line* frames.

	Initial and final range (unit: feet)	Absolute errors (unit: feet)	Relative errors (unit: %)	# frames for 10% range estimate error
Truck 1 (right most)	(272.7, 176.1)	(5.1, 0.1, 3.8)	(2.7, 0.03, 2.0)	5
Truck 2 (left most)	(365.5, 268.9)	(5.1, 0.6, 2.0)	(1.8, 0.2, 0.7)	20
Truck 3 (between center and right most)	(503.0, 406.4)	(12.2, 0.5, 1.2)	(2.8, 0.1, 0.3)	30
Truck 4 (between center and left most)	(615.9, 519.3)	(37.3, 6.0, 0.3)	(7.0, 1.1, 0.1)	70
Truck 5 (center)	(771.8, 675.2)	(83.7, 0.4, 0.5)	(12.4, 0.1, 0.1)	> 90

Table II Position estimates of various trucks after processing 90 *arc* frames. (Truck 1 is not visible in all frames).

	Initial and final range (unit: feet)	Absolute errors (unit: feet)	Relative errors (unit: %)	# frames for 10% range estimate error
Truck 2 (left most)	(224.9, 107.5)	(2.6, 1.3, 0.6)	(1.7, 0.8, 0.4)	20
Truck 3 (right most)	(384.6, 267.2)	(4.1, 1.9, 1.7)	(1.5, 0.7, 0.6)	20
Truck 4 (second from left)	(475.3, 357.9)	(90.0, 19.2, 0.6)	(24.5, 5.2, 0.2)	> 90
Truck 5 (second from right)	(631.2, 513.8)	(79.6, 3.7, 1.3)	(14.9, 0.7, 0.2)	> 90

5 Conclusion

In this paper, we proposed a motion estimation system which is capable of accurately estimating the 3D object positions in the scene. We used the piece-wise linear approximation for the flightpath to facilitate the construction of the epipolar planes and lines. This is very suitable for rotorcrafts having on-board inertial navigation systems, where the camera states are always available. The piece-wise linear approximation turns out to be a good method to solve the difficult motion problem resulting from general 3D camera motion. We also built an efficient feature tracker which makes use of the epipolar constraint to achieve good feature matching results. Weighted incremental least squares estimator then performs the position estimation. The final relative error of the range estimate for the object approximately 176 feet away (truck 1 in the *line* sequence) is less than 3%, which corresponds to only 5 feet absolute error. In addition, the range estimation for truck 1 takes less than 10 frames to converge within 10% error. Since the speed of the helicopter is 35 feet/sec in the *line* sequence, this gives the pilot about 8 seconds to avoid the obstacle. For the estimation on other trucks, the pilot actually has ample time to make decision about the flightpath. Currently, our algorithms run at the rate of two frames per second on a DECstation 5000/240 for tracking several hundreds of features on 512×512 image sequences (excluding edge detection). However, optimization for throughput was not a consideration in this work.

Reference

[Baker89] Baker, H.H. and R.C. Bolles, "Generalizing Epipolar-Plane Image Analysis on the Spatiotemporal Surface," *Int. J. Computer Vision*, Vol. 3, pp. 33–49, 1989.

[Bhanu89] Bhanu, B., B. Roberts, and J.C. Ming, "Inertial Navigation Sensor Integrated Motion Analysis," *Proc. DARPA Image Understanding Workshop*, pp. 747–763, 1989.

[Bolles87] Bolles, R.C, H.H. Baker, and D.H. Marimont, "Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion," *Int. J. Computer Vision*, Vol. 1, pp. 7–55, 1987.

[Canny86] Canny, J., "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679–698, 1986.

[Cheng91] Cheng, V.H.L. and B. Sridhar, "Considerations for Automated Nap-of-the-Earth Rotorcraft Flight," *Journal of the American Helicopter Society*, Vol. 36, No. 2, pp. 61–69, 1991.

[Cox93] Cox, I.J., "A Review of Statistical Data Association Techniques for Motion Correspondence," *Int. J. Computer Vision*, Vol. 10, No. 1, pp. 53–66, 1993.

[Crowley88] Crowley, J.L., P. Stelmaszyk, and C. Discours, "Measuring Image Flow by Track Edge-lines," *Proc. Int. Conf. Computer Vision*, pp. 658–664, 1988.

[Matthies89] Matthies, L., T. Kanade, and R. Szeliski, "Kalman Filter-based Algorithms for Estimating Depth from Image Sequences," *Int. J. Computer Vision*, Vol. 3, No. 3, pp. 209–236, 1989.

[Morefield77] Morefield, C.L., "Application of 0-1 Integer Programming to Multitarget Tracking Problems," *IEEE Trans. Automatic Control*, AC-22(6), 1977.

[Roberts91] Roberts, B., B. Sridhar, and B. Bhanu, "Inertial Navigation Sensor Integrated Motion Analysis for Obstacle Detection," *Digital Avionics Systems Conference*, pp. 131–136, 1991.

[Sawhney93] Sawhney, H.S., J. Oleinsis, and A.R. Hanson, "Image Description and 3-D Reconstruction from Image Trajectories of Rotational Motion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 9, pp. 885–898, 1993.

[Smith75] Smith, P. and G. Buechler, "A Branching Algorithm for Discriminating and Tracking Multiple Objects," *IEEE Trans. Automatic Control*, AC-20: pp. 101–104, 1975.

[Smith90] Smith, P.N., "NASA Image Data Base User's Guide," NASA Ames Research Center, Moffett Field, CA., Version 1.0, 1990.

[Smith92] Smith, P.N., B. Sridhar, and B. Hussien, "Vision-Based Range Estimation Using Helicopter Flight Data," *IEEE Conf. Computer Vision and Pattern Recognition*, pp. 202–208, 1992.

[Sridhar89] Sridhar, B., V.H.L. Cheng, and A.V. Phatak, "Kalman Filter Based Range Estimation for Autonomous Navigation Using Imaging Sensors," *Proc. 11th IFAC Symposium on Automatic Control in Aerospace*, 1989.

[Therrien89] Therrien, C.W., *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*, Wiley, New York, 1989.

Image Processing and Data Classification

PRECEDING PAGE BLANK NOT FILMED