5-2009

Object Trajectory Estimation Using Optical Flow

Shuo Liu
Utah State University

OBJECT TRAJECTORY ESTIMATION USING OPTICAL FLOW

by

Shuo Liu

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Prof. Paul Israelsen Major Professor	Dr. Robert T. Pack Committee Member
Dr. Aravind Dasu Committee Member	Dr. Byron R. Burnham Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2009

Abstract

Object Trajectory Estimation Using Optical Flow

by

Shuo Liu, Master of Science

Utah State University, 2009

Major Professor: Prof. Paul Israelsen
Department: Electrical and Computer Engineering

Object trajectory tracking is an important topic in many different areas. It is widely used in robot technology, traffic, movie industry, and others. Optical flow is a useful method in the object tracking branch and it can calculate the motion of each pixel between two frames, and thus it provides a possible way to get the trajectory of objects. There are numerous papers describing the implementation of optical flow. Some results are acceptable, but in many projects, there are limitations. In most previous applications, because the camera is usually static, it is easy to apply optical flow to identify the moving targets in a scene and get their trajectories. When the camera moves, a global motion will be added to the local motion, which complicates the issue. In this thesis we use a combination of optical flow and image correlation to deal with this problem, and have good experimental results. For trajectory estimation, we incorporate a Kalman Filter with the optical flow. Not only can we smooth the motion history, but we can also estimate the motion into the next frame. The addition of a spatial-temporal filter improves the results in our later process.

(63 pages)

Acknowledgments

I would like to thank my advisor, Prof. Paul Israelsen, my committee members, Dr. Robert T. Pack and Dr. Aravind Dasu, for investing in me and for their helpful guidance throughout my research.

I would like to thank the people in the Space Dynamics Lab for their help. I would also acknowledge Adam Fowles for his important contributions to this project at the very beginning.

Last, but not least, I would like to thank my parents, Liu, Sancheng and Liu, Shuqin, for their forever support.

Shuo Liu

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Object trajectory is a key computer vision topic that aims at detecting the position of a moving object in a video sequence [1]. It can be widely used in robotics, traffic, and the movie technology fields [2]. After several people published optical flow algorithms in the late 19th century, object trajectory tracking began to receive new interest.

During that period, numerous projects and experiments were done which gave acceptable results. One of the most popular usages of optical flow is object trajectory tracking. A camera or an observer is fixed somewhere and takes a continuous video or frames of the object. The observer is immobile so records a static background. The object has a relative motion with the background. By running optical flow algorithms, it is possible to determine the approximate relative motion of the object.

The ultimate aim for our project is to implement the optical flow theory into photogrammetry. In photogrammetry, there is a step called "triangulation." Triangulation is the principle used by photogrammetry to produce 3-dimensional point measurements. By mathematically intersecting converging lines in space, the precise location of the point can be determined. Photogrammetry can measure multiple points at a time with virtually no limit on the number of simultaneously triangulated points. Basically, the configuration consists of two sensors observing the item. Or a single sensor observing the item from two different points in space. The projection centers and the considered interested point define a triangle spatially. If the distance between the sensors (called the base) is known, by determining the angles between the projection rays of the sensors, and the intersection point, the 3-dimensional coordinate could be calculated from the triangular relations. By adding optical flow into photogrammetry, we do not need so many points tested. We could use the optical flow to simulate observation points. What is more, in the field of photogrammetry, the more observation points we have, the higher accuracy we can achieve. optical flow could be used to produce more points if we want to improve the locating accuracy [2, 3].

In this thesis, some basic optical flow and image correlation algorithms will be introduced. The techniques for using them and the comparisons of the results will be given separately. There will be also the discussion about the limitation of optical flow, the difficulties in our project, and the solutions.

Chapter 2
Optical Flow

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer, such as an eye or a camera, and the scene. It is the distribution of apparent velocities of brightness pattern movement in an image. Optical flow can arise from relative motion of objects and viewer, so it could give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement [4]. This theory could be used in motion detection, object segmentation, motion compensation, and stereo disparity measurement.

Optical flow cannot be computed locally, some assumed constraints will be given when the algorithms are explained.

2.1 Optical Flow Algorithms

In the optical flow field, there are two basic and classic algorithms. Both of them were invented in 1980's, and they are easy to be implemented.

2.1.1 The Lucas & Kanade Algorithm

The Lucas & Kanade algorithm is a solution of image registration [5]. Image registration has a variety of applications in computer vision, such as, image matching for stereo vision, pattern recognition, and motion analysis. Existing techniques for image registration tend to be costly, and they fail to deal with image rotation and distortions. The Lucas & Kanade algotithm presented a new method that uses spatial intensity gradient information to direct the search for the position that yields the best match. This is how the Lucas & Kanade algorithm came about [5–7].

Between two adjacent frames, there will be δx and δy displacements during time δt.

Under the assumption that the intensity of an object remains constant, we can find the following equation:

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t). \tag{2.1}$$

If the motion during δt is small enough, we could expand $I(x, y, t)$ into a Taylor series to get

$$I(x + x, y + y, t + t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + H.O.T, \tag{2.2}$$

where H.O.T are the higher order terms and can be ignored.

Because of the constraint,

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0, \tag{2.3}$$

use V_x, V_y to represent the velocity or optical flow of $I(x, y, t)$ in x and y direction, respectively. $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, and $\frac{\partial I}{\partial t}$ $(I_x, I_y,\ and\ I_t)$, are the partial derivatives of the image at (x, y, t) with respect to the corresponding variables.

Thus,

$$I_x V_x + I_y V_y = -I_t, \tag{2.4}$$

there are two unknowns but just one equation. We need another constraint to build one more equation. The additional constraint for the Lucas & Kanade algorithm is to assume that the optical flow is locally constant. That is the optical flow (V_x, V_y) is constant in a small window, which is centered at the point (x, y). The pixels around it are indexed as 1 . . . n, then

$$\begin{aligned} I_{x1} V_x + I_{y1} V_y &= -I_{t1}, \\ I_{x2} V_x + I_{y2} V_y &= -I_{t2}, \\ &\vdots \\ I_{xn} V_x + I_{yn} V_y &= -I_{tn}. \end{aligned} \tag{2.5}$$

The number of equations is larger than the number of unknowns. It is now an over-determined system,

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}, \tag{2.6}$$

or

$$A\vec{V} = -b. \tag{2.7}$$

We use the least squares method that minimizes $\|A\vec{V} + b\|^2$ and get

$$\vec{V} = (-A^T A)^{-1} A^T b, \tag{2.8}$$

or

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = - \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}^{-1} (A^T I_t). \tag{2.9}$$

2.1.2 The Horn - Schunck Algorithm

To avoid variations in brightness due to shading effects, objects are assumed to have flat surfaces. The illumination across the surface is assumed to be uniform. Horn and Schunck assumed that reflectance varies smoothly and has no spatial discontinuities. This assures them that the image brightness is differentiable.

First, they derive an equation that relates the change in image brightness at a point to the motion of the brightness pattern. $E(x, y, t)$ represents the image brightness at the point (x, y) at time t. The brightness of a particular point in the pattern is constant, so that

$$\frac{dE}{dt} = 0, \tag{2.10}$$

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0. \tag{2.11}$$

We will let

$$u = \frac{dx}{dt} \ \mathit{and} \ v = \frac{dy}{dt}. \tag{2.12}$$

We then get one signal linear equation which has two unknown parameters u and v.

$$E_x u + E_y v + E_t = 0, \tag{2.13}$$

an additional equation is needed.

If every point of the brightness pattern can move independently, there is little hope to recover the velocities [4]. We will assume that the neighboring points on the objects have similar velocities and the velocity field of the brightness patterns in the image varies smoothly. This constraint can be expressed by limiting the difference between the flow velocity at a point and the average velocity over a small neighborhood containing the point. Equivalently we can minimize the sum of the squares of the Laplacians of the x and y components of the flow. The Laplacians of u and v:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \ \mathit{and} \ \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \tag{2.14}$$

In simple situations, both Laplacians are zero.

In order to estimate the derivatives of brightness from the discrete set of image brightness measurements available, Horn and Schunck use a set which gives estimate of E_x,E_y,E_t at a point in the center of a cube formed by eight measurements. Each of the estimates is the average of four first differences taken over adjacent measurements in the cube [4].

$$\begin{aligned}
E_x &\approx \frac{1}{4}\{E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} + \\
&E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\} \\
E_y &\approx \frac{1}{4}\{E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} + \\
&E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\} \\
E_t &\approx \frac{1}{4}\{E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} + \\
&E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\}
\end{aligned} \tag{2.15}$$

The unit of the length in the reference is the grid spacing interval in each image frame, and the unit of time is the image sampling period (fig. 2.1).

The Laplacians of u and v also should be approximated.

$$\nabla^2 u \approx k(\vec{u}_{i,j,k} - u_{i,j,k}) \ and \ \nabla^2 v \approx k(\vec{v}_{i,j,k} - v_{i,j,k}) \tag{2.16}$$

The proportionality factor k here equals 3. The local averages $\overline{u}$ and $\overline{v}$ are defined as follows (fig. 2.2):

$$\begin{aligned}
\vec{u}_{i,j,k} &= \frac{1}{6}\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\} \\
+\frac{1}{12}&\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\} \\
\vec{v}_{i,j,k} &= \frac{1}{6}\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\} \\
+\frac{1}{12}&\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\}.
\end{aligned} \tag{2.17}$$

The sum of the total errors is minimized

$$\varepsilon^2 = a^2{\varepsilon_c}^2 + {\varepsilon_b}^2. \tag{2.18}$$

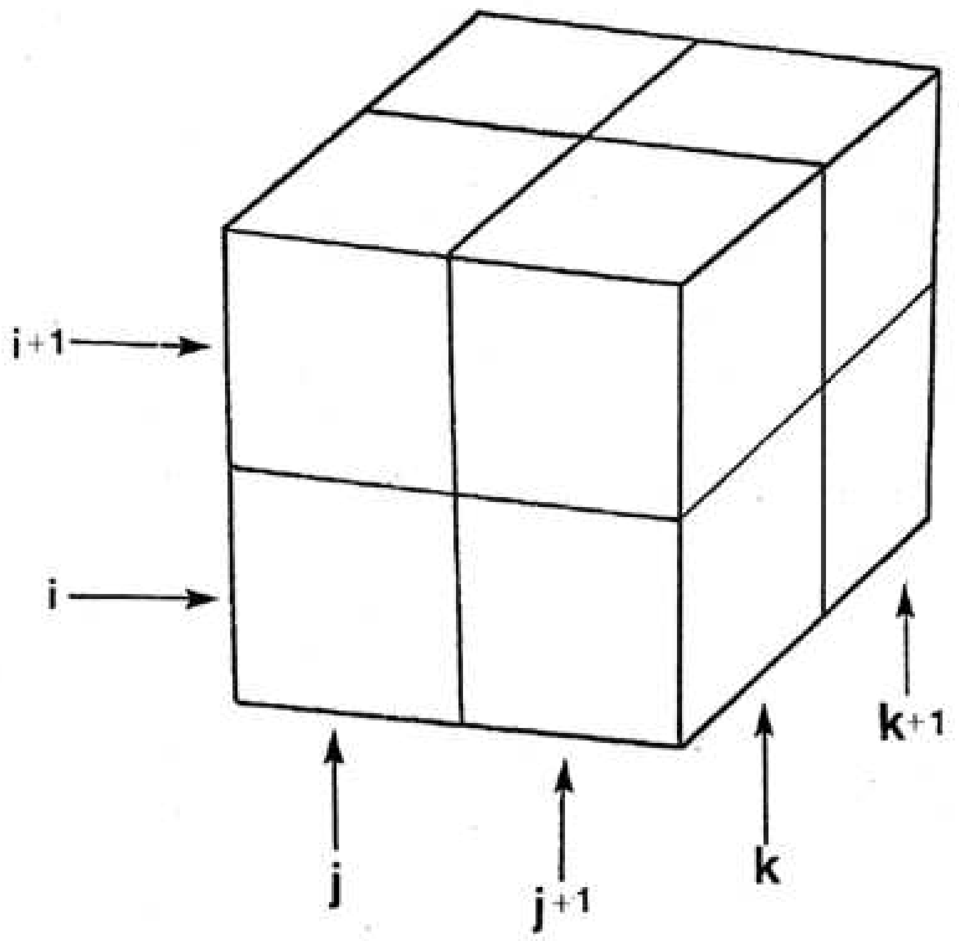


Fig. 2.1: The three partial derivatives of image brightness at the center of the cube are each estimated from the average of first differences along four parallel edges of the cube. Here the column index j corresponds to the x direction in the image, the row index i to the y direction, while k lies in the time direction.

The a^2 is a weighting factor. Image brightness measurements may be corrupted by quantization error and noise so we cannot expect ϵ_b to be identically zero. This quantity will tend to have an error magnitude that is proportional to the noise in the measurement. That is why a^2 is chosen.

Where

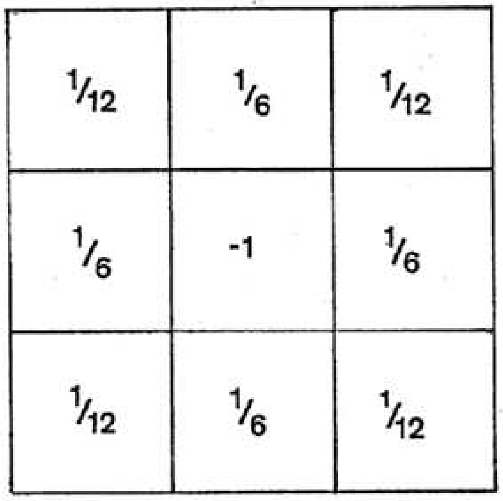
$$\epsilon_b = E_x u + E_y v + E_t, \tag{2.19}$$

$$\epsilon_c{}^2 = (\vec{u} - u)^2 + (\vec{v} - v)^2, \tag{2.20}$$

iterative solution

$$u^{n+1} = \overline{u}^n - \frac{E_x[E_x\overline{u}^n + E_y\overline{v}^n + E_t]}{(a^2 + E_x{}^2 + E_y{}^2)}, \tag{2.21}$$

$$v^{n+1} = \overline{v}^n - \frac{E_y[E_x\overline{u}^n + E_y\overline{v}^n + E_t]}{(a^2 + E_x{}^2 + E_y{}^2)}. \tag{2.22}$$



$1/12$	$1/6$	$1/12$
$1/6$	-1	$1/6$
$1/12$	$1/6$	$1/12$

Fig. 2.2: The Laplacian is estimated by subtracting the value at a point from a weighted average of the values at neighboring points.

2.2 Testing of Optical Flow

As just mentioned, optical flow is the pattern of apparent motion of objects caused by relative motion between objects and viewers, we can use optical flow to identify a moving object in a scene easily.

In order to test the effect of optical flow, we ran it on a video sequence which had be acquired from a fixed wing aircraft.

In fig. 2.3 and fig. 2.4 we can see that optical flow can show numerical motions with directions. Almost all the moving cars in this frame and its adjacent next frame are identified clearly. In the two objects which are circled out, because of their different characteristics, the optical flow data gives two different results. The upper car is nearly blurred by the noise around it. The accuracy of optical flow is affected by many factors.

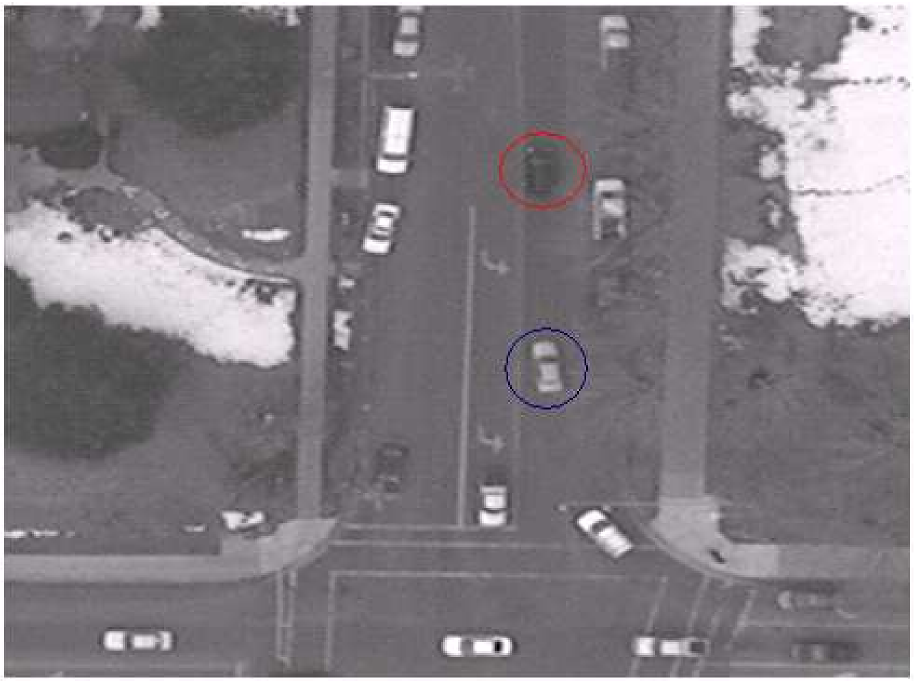


Fig. 2.3: Two cars of different characteristics.

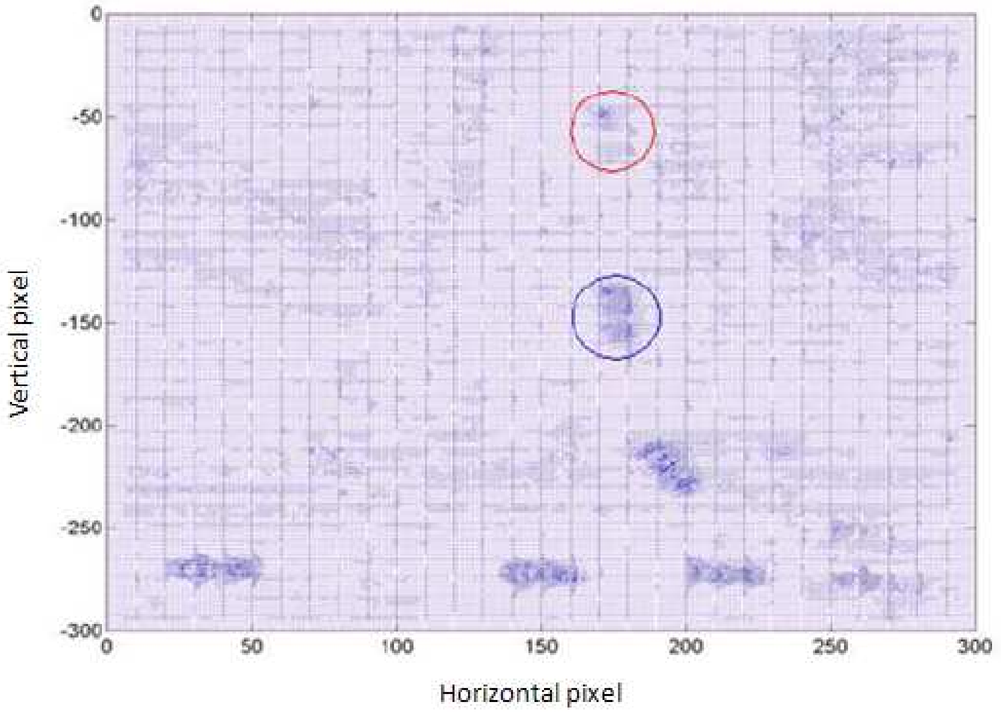


Fig. 2.4: Local motions of the two different cars.

Chapter 3
Global Motion

3.1 Image Correlation

The correlation between two signals is a standard approach for feature detection. It is an important component in our system to detect the global motion. We selected the NCC (normalized cross-correlation), one of the various methods to calculate the global motion [8,9].

In feature tracking approaches, there are a number of different algorithms, for example, the SSDA (sequential similarity detection algorithm), gradient descent search, snakes, etc. Each of these algorithms has its advantages and disadvantages. A previous study of the different algorithms in the presence of various image distortions found that NCC provides the best performance in all image categories, even though it is not perfect. It makes few requirements on the image sequence and has no parameters to be searched by the user [8,9].

Cross correlation is a standard method of estimating the degree to which two series are correlated. Consider two series $x(i)$ and $y(i)$, where $i = 0, 1, 2, ..., N-1$. The cross correlation r at delay d is defined as

$$r = \frac{\sum_i [x(i) - \bar{x}][y(i) - \bar{y}]}{\sqrt{\sum_i [x(i) - \bar{x}] \sum_i [y(i) - \bar{y}]}}, \tag{3.1}$$

where $\bar{x}$ and $\bar{y}$ are the means of the corresponding series.

3.2 Jitter Removal

We used the cross correlation algorithm to detect global motion in a video sequence in order to remove aircraft motion and jitter.

We have many aerial video sequences. A plane in flight is easily affected by wind and turbulence. We would like to remove these motions from our video sequence to represent only the linear motion of the aircraft. First, we need to calculate the displacement of each two adjacent frames. We ran the NCC image correlation algorithm on the original frames from the aerial video sequence to get the displacements, add them together to reflect the whole flight. On the summation of the displacements, we use least square method to synthesis a new straight flight line. From the difference between the new synthesized line and the summation of the displacements, we can know by moving how many pixels and in which direction to correct the original flight line. In fig. 3.1, the blue curve shows the original summation of displacements of the each two adjacent frames. The red line is the linear flight path of the aircraft. The green curve shows the difference between the blue curve and the red line. It signifies the number of pixels by which we should displace each video frame in order to remove swaying and jitter.

After doing the experiment, we still wanted to check about how was the effect of the modification. We ran image correlation again on the modified sequence to get the displacements between two adjacent frames, and calculated out the summation of the displacements, synthesized a "modified" line again based on the modified sequence.

In fig. 3.2, the blue curve is the modified sequence flight line. Red line is the synthesized flight line based on the modified sequence. The green curve tells the difference between the blue curve and the red line. The blue curve keeps almost the same shape as the previous one, but the amplitude has decreased significantly. That means the algorithm works well for our aerial video sequence. If we use it iteratively, the experimental result can be improved.

3.3 Optical Flow with Global Motion Removal

3.3.1 Global Motion Removal

When we studied the previous optical flow applications, one problem became more and more obvious. Most previous experiments used a static viewer to test or record a static background. Only the objects in the scene moved. In this situation, it is easy to identify the moving parts in a scene just by running optical flow directly. In our projects, we have many aerial videos. The camera is installed on the plane and it moves together with the plane. In this circumstance, both the viewer and background move at the same time. If we run optical flow directly on the original sequence, it is not easy for us to find the moving parts in the background. Our project provides a good method to solve this problem.

Figure. 3.3 shows four adjacent frames extracted from an aerial sequence. The frames contain some moving targets. After running optical flow, it is difficult to identify the moving targets in these sequences, as shown in fig. 3.4. The moving objects are blurred by the noise around them. In fact, the moving objects are also blurred by the global motion. The global motion is larger than the object local motion, so when running optical flow, the data shows the global motion instead of showing the local motion.

As image correlation can remove jitter, it can be used to remove the global motion in our video sequences. Optical flow works after global motion removal. The result is shown in fig. 3.5. The global motion removal helps making the object detection easier [10, 11].

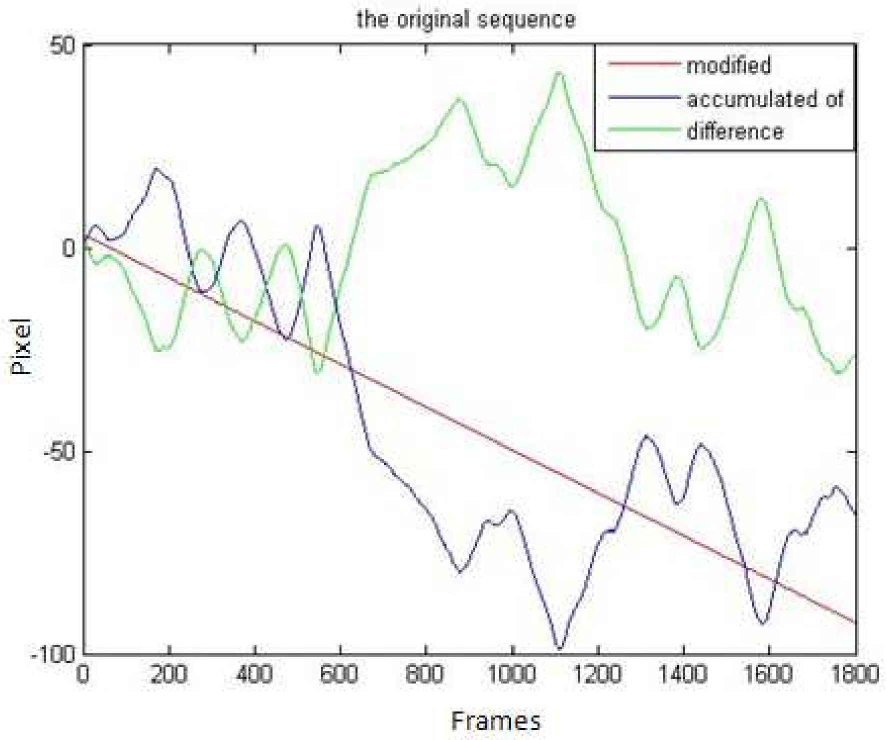


Fig. 3.1: Flight line modification.

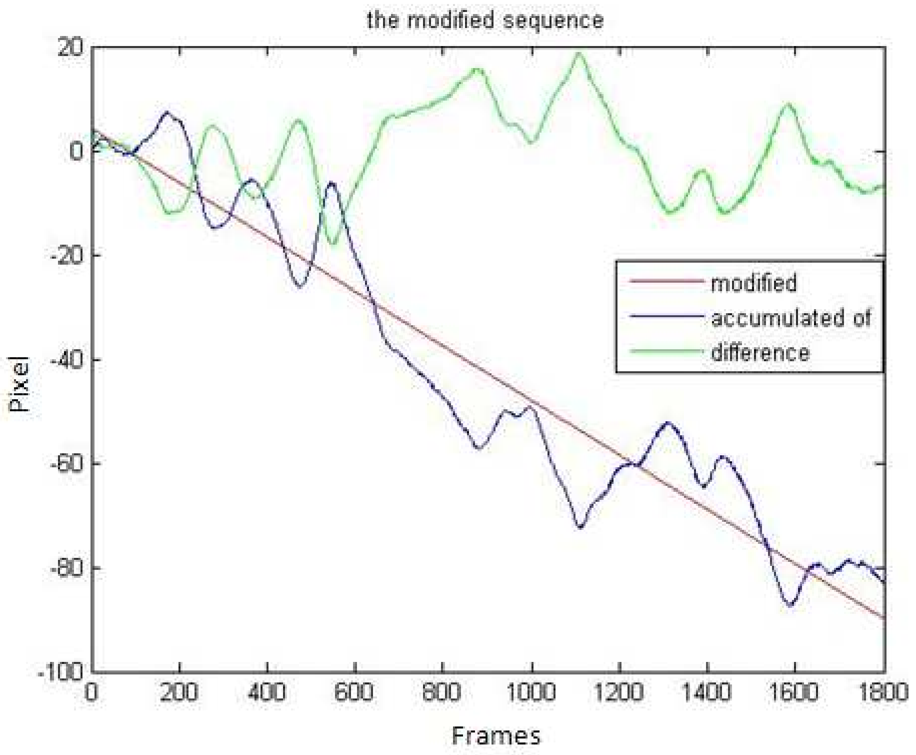


Fig. 3.2: Flight line modification.

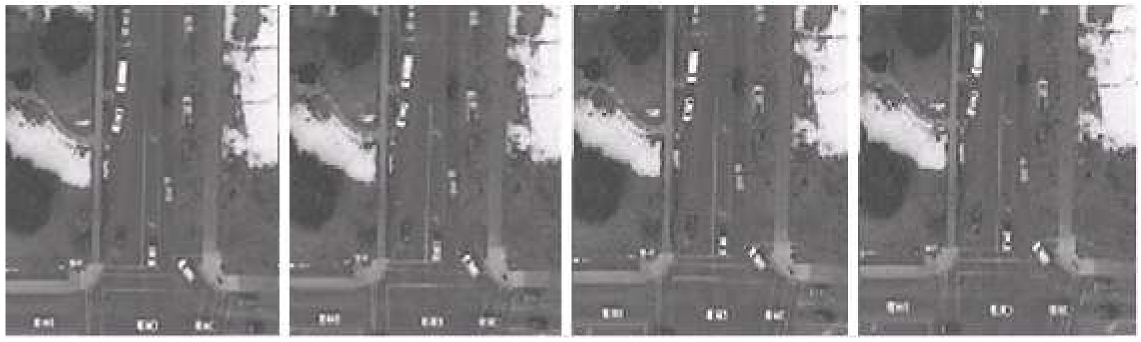


Fig. 3.3: Four adjacent frames.

3.3.2 Comparison Between the Lucas & Kanade and the Horn - Schunck

Different optical flow algorithms may produce different results on the same image data. In some previous work, the Lucas & Kanade is shown to be more robust if the image contains noise, and the Horn - Schunck is a method which could provide us with dense results (dense here means the algorithm can calculate the motion at each pixel location at any point in the sequence) [12,13]. It is hard to say which way is ideal. The results can help us to determine which method should be used [4–7,14–22].

The results of running each algorithm after global motion removal are shown in figs. 3.6 and 3.7. The results show that the Horn - Schunck algorithm provides better results than the Lucas & Kanade did for this video. (The code is shown in Appendix A.)

However, it does not mean the Horn - Schunck is always superior. Our experience has shown that we should try both of them first, evaluate the results, and then decide which one should be used. Other optical flow algorithms have also been evaluated, but due to the ease of the implementation of these two algorithms and the better results they achieved, they were used in our further research. If we set a threshold for the two algorithms to delete the redundancy vectors, the results can be clearer, as shown in figs. 3.8 and 3.9. The threshold could be set according to our experiences. For this video, we deleted all the data which indicated the pixels moved in different directions. For example, if we want to track car A, we find the main direction of the car and delete the other disturbing directions. After that, just the correct information is left in the frames.

The results shown above makes us confident to use the combination of global motion removal and Horn - Schunck in our car detecting and tracking experiment.

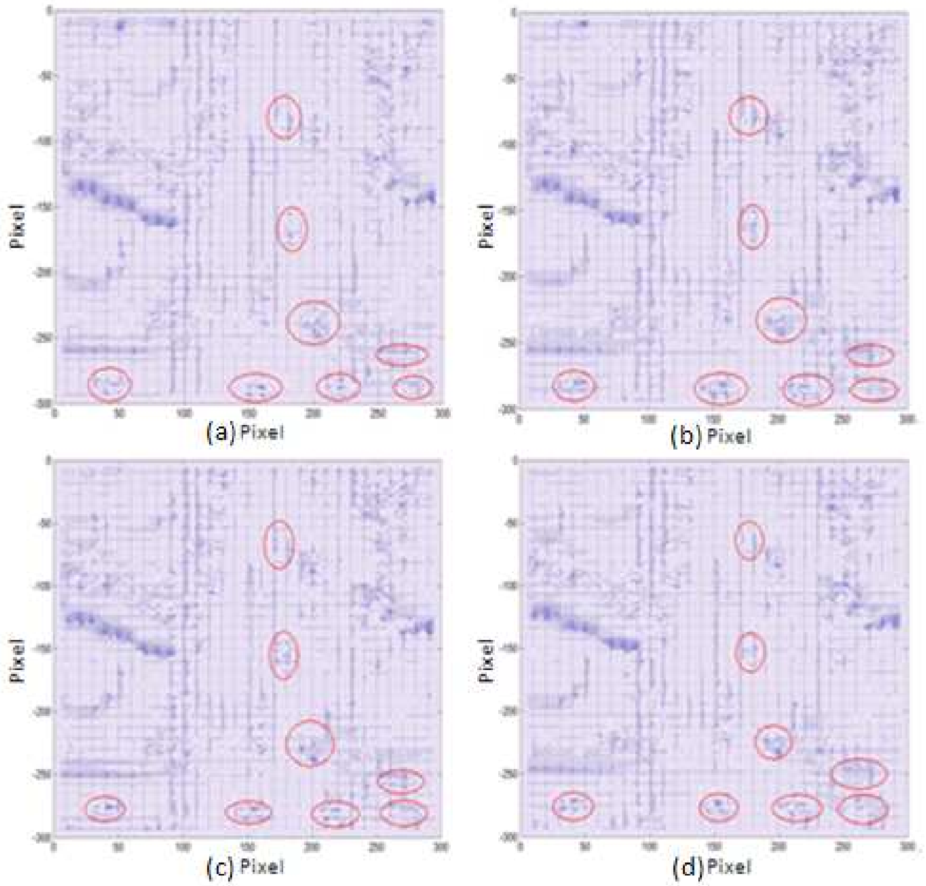


Fig. 3.4: Moving object detection without deleting global motion (a, b, c, d are four adjacent frames from the same sequence).

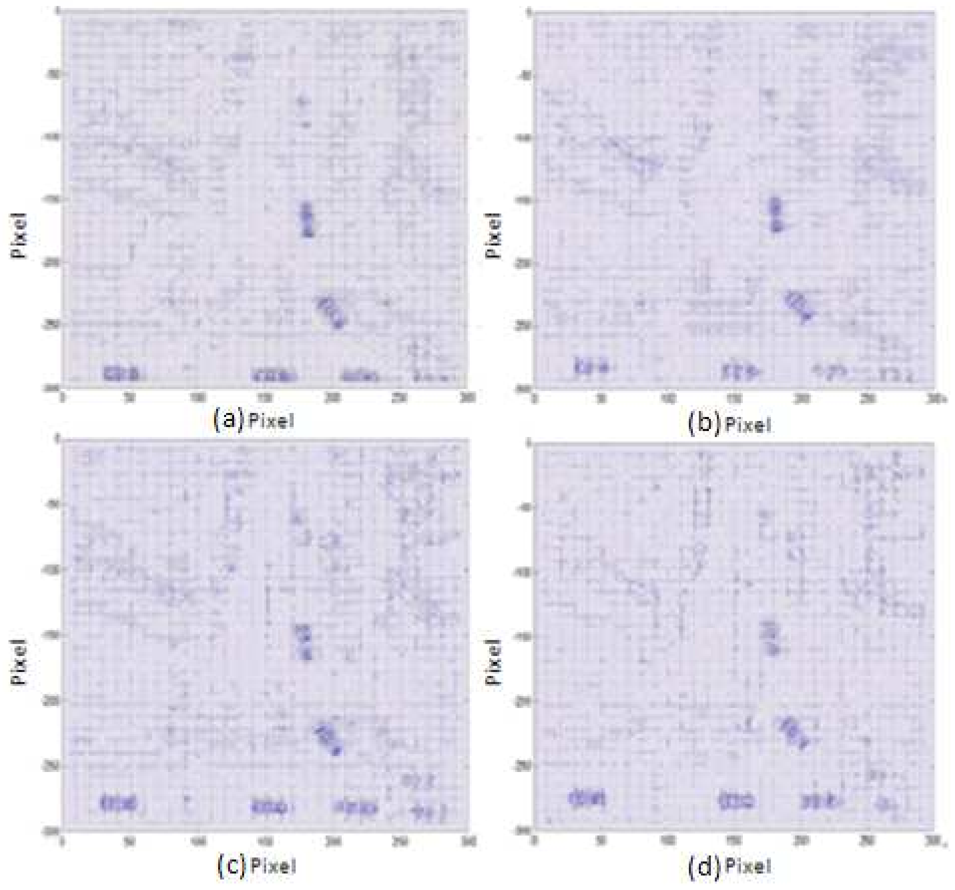


Fig. 3.5: Moving object detection with deleting global motion (a, b, c, d are four adjacent frames from the same sequence).

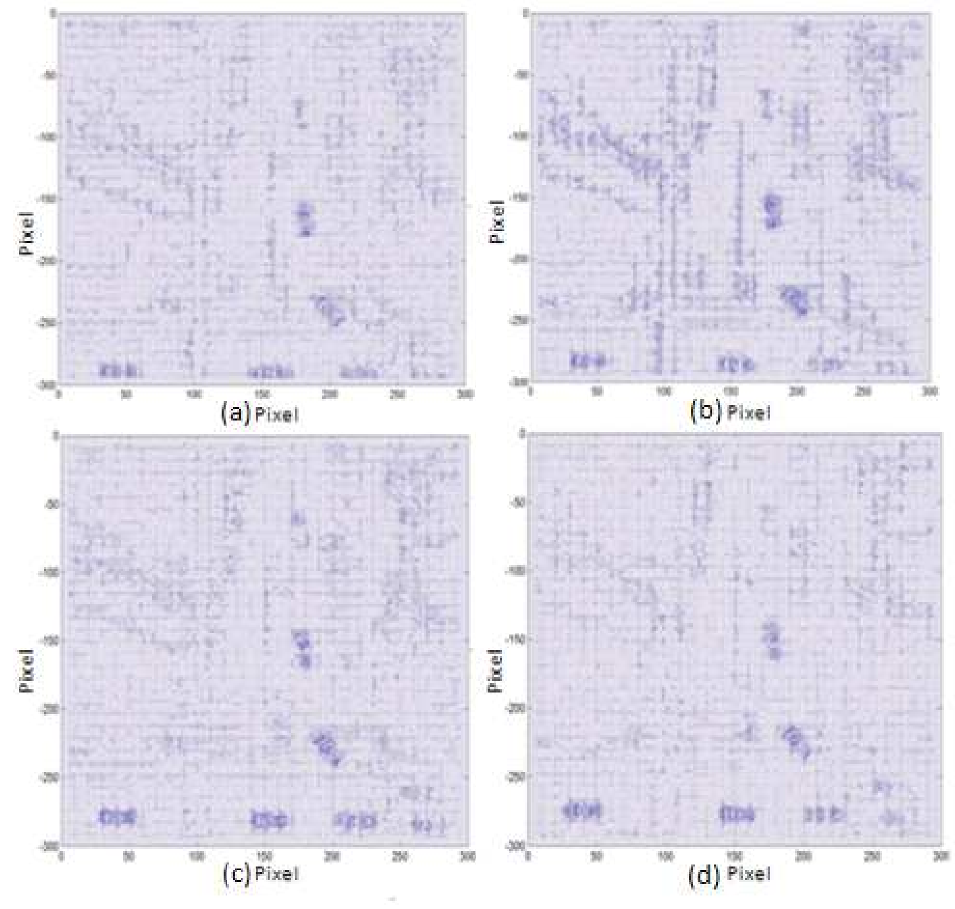


Fig. 3.6: Local motion gotten by using the Horn - Schunck (a, b, c, d are four adjacent frames from the same sequence).

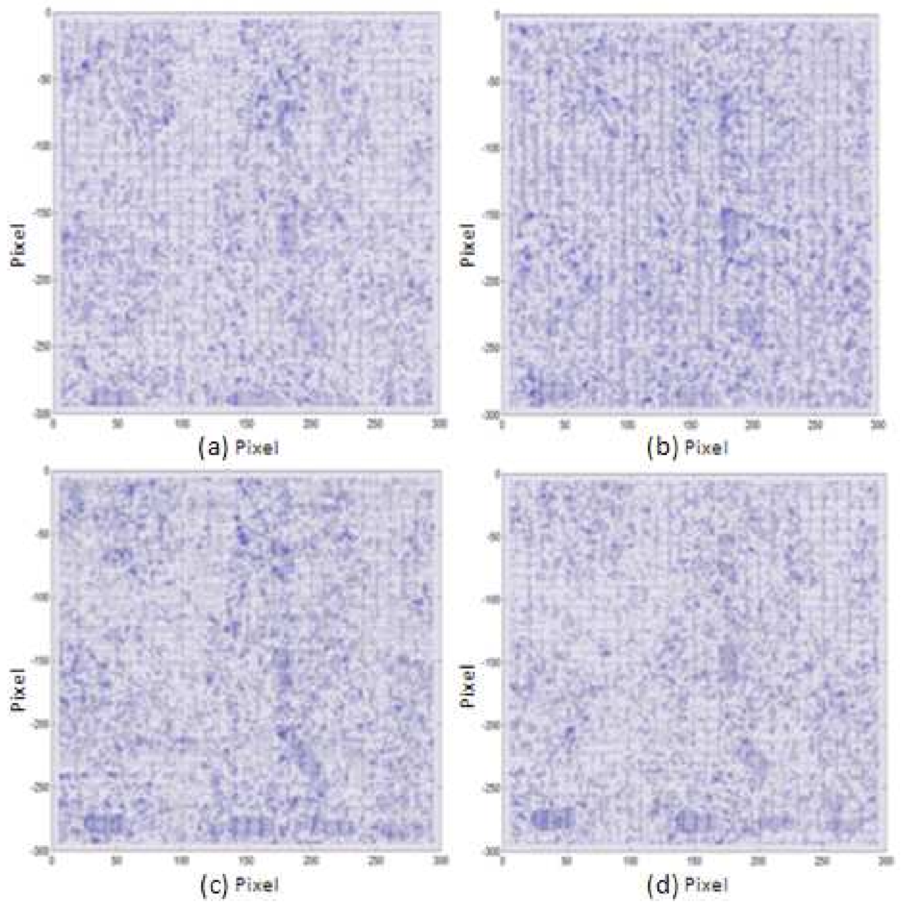


Fig. 3.7: Local motion gotten by the Lucas & Kanade (a, b, c, d are four adjacent frames from the same sequence).

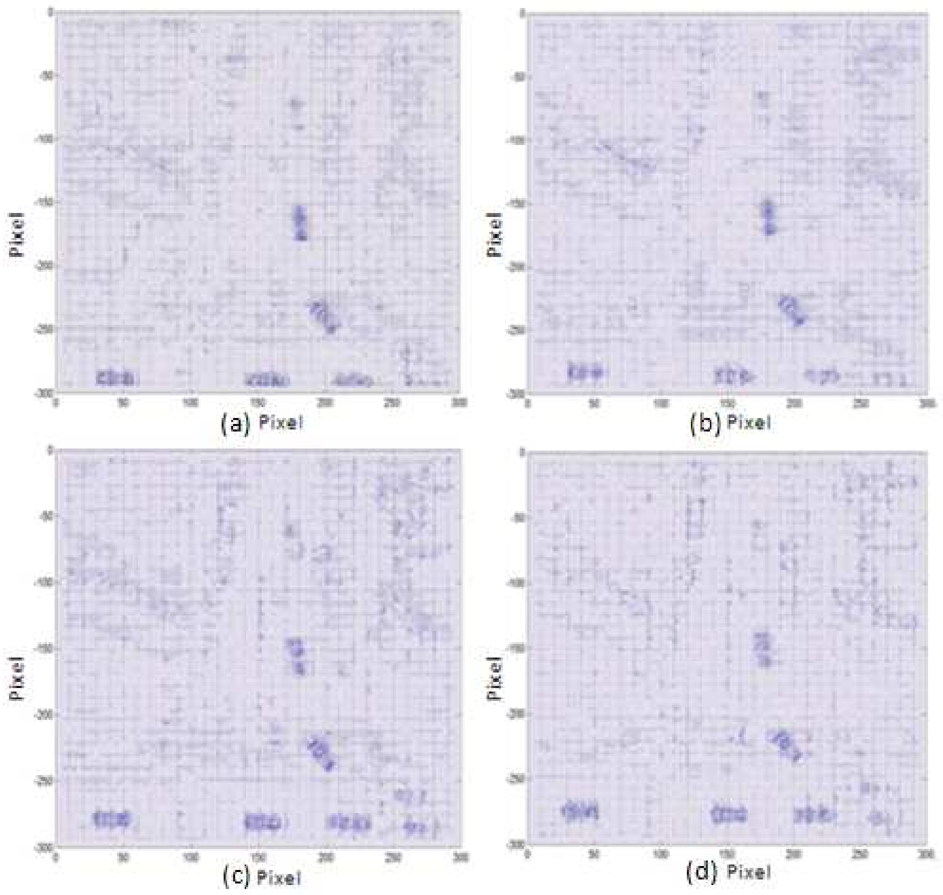


Fig. 3.8: Local motion gotten by the Horn - Schunck with threshold (a, b, c, d are four adjacent frames from the same sequence).

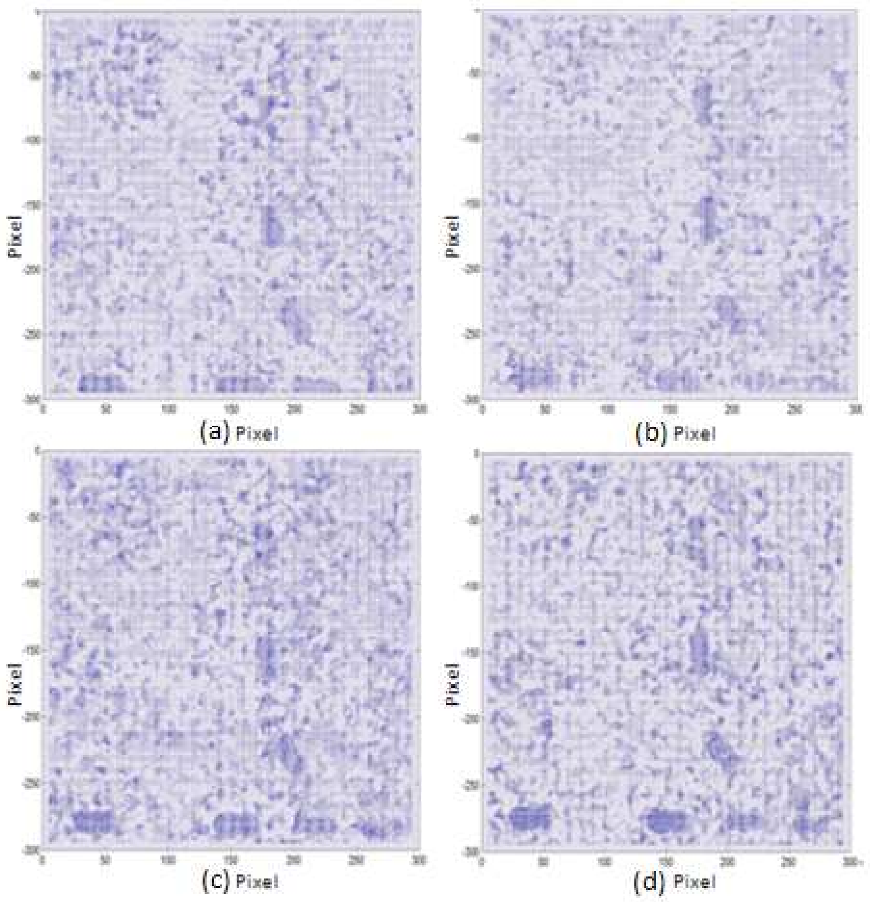


Fig. 3.9: Local motion gotten by the Lucas & Kanade with threshold (a, b, c, d are four adjacent frames from the same sequence).

Chapter 4
Object Tracking

As mentioned in introduction of this thesis, we want to apply Optical Flow to photogrammetry. We want to use the motion history and current motion information to estimate where the object or a particular point will move to in the next adjacent frame.

The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of the video [23]. The tasks of detecting the object and establishing correspondence between the object instances across frames can either be performed separately or jointly. If the tasks are performed separately, possible object regions in every frame are obtained by means of an object detection algorithm, and then the tracker gives the trajectory of objects across frames. If the tasks are performed jointly, the object region and correspondence is jointly estimated by iteratively updating the object location and region information obtained from previous frames. We used the latter technique in our experiments.

Among the several tracking methods (as shown in fig. 4.1), point tracking is easy. Objects detected in consecutive frames are represented by points, and the association of the points is based on the previous object state which can include object position and motion. We first use image correlation to determine the global motion, and subtract the global motion for each pixel in the image so only local motion remains. We ran the optical flow on the modified frames and calculated the motion. This motion data is stored for tracking, requiring that we store all the Optical Flow data of all the pixels in each frame. Having the data for all the pixels in each frame ensures that we can check each pixel's motion in each frame. For example, some times the objects we are tracking will move out of the frame's edge or be occluded by other objects, and it is meaningless to keep tracking them. We then can select other objects to track at any time [3, 23, 24].

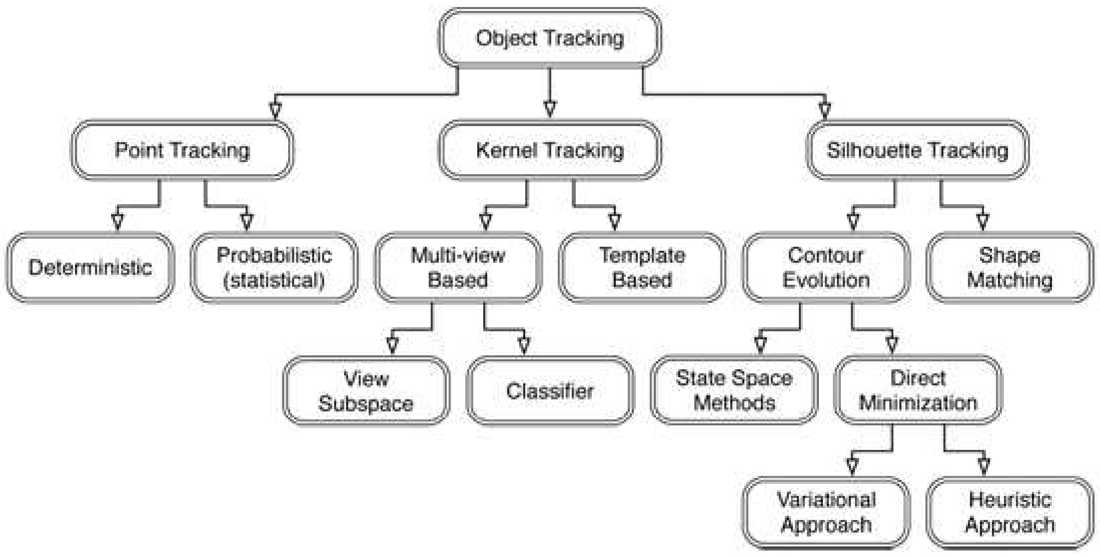


Fig. 4.1: Taxonomy of tracking methods.

4.1 Kalman Filter

In order to more accurately estimate motion trajectories we incorporated a Kalman Filter into our project. The Kalman Filter is an efficient recursive filter that can estimate the next state of a dynamic system based on noisy measurement. One advantages of Kalman Filter is that it does not need all the motion history to estimate the next stage motion, it just needs the nearest one from the current motion [25].

Here we will explain the Kalman Filter we used. In the implementation of Kalman Filter, we have five equations

$$X(k \mid k-1) = AX(k-1 \mid k-1) + BU(k) + W(k), \tag{4.1}$$

where the $X(k \mid k-1)$ is the estimated system state based on the previous optimized state, $U(k)$ is the control element, if there is no control element, it could be 0.

Now the system has been updated, but the covariance still needs to be updated. We use P to indicate the covariance:

$$P(k \mid k-1) = AP(k-1 \mid k-1)A' + Q. \tag{4.2}$$

in the eq. (4.2), $P(k \mid k-1)$ is the covariance of $X(k \mid k-1)$, $P(k-1 \mid k-1)$ is the covariance of $X(k-1 \mid k-1)$, Q is the process noise covariance. We need to collect the testing data. The optimized state estimation

$$X(k \mid k) = X(k \mid k-1) + Kg(k)(Z(k) - HX(k \mid k-1)). \tag{4.3}$$

The Kg here means the Kalman gain

$$Kg(k) = \frac{P(k \mid k-1)H'}{HP(k \mid k-1)H' + R}. \tag{4.4}$$

We have the optimized system estimation $X(k \mid k)$ now. The covariance of $X(k \mid k)$ is still needed in order to do the future state estimation.

$$P(k \mid k) = (1 - Kg(k)H)P(k \mid k-1) \tag{4.5}$$

We have no control element in our system, so the $U(k)$ in eq. (4.1) is 0. In our project, all the Optical Flow data are stored in txt format files. This makes it easy for us to reuse them later. If there is no local motion, we assume the next state estimation $X(k \mid k-1)$ equals to the previous optimized state estimation $X(k-1 \mid k-1)$, and the A in eq. (4.1) is 1. The Optical Flow data will be read out and perform as the testing data, the H in eq. (4.3) is 1.

We add the Kalman Filter into a single pixel, get an optimized motion estimation for this pixel. We track the same single pixel of an object all the time, which means we use the image correlation data to track the pixel. This ensures that we have the optimized motion estimation for the same pixel (errors contained in this process).

All the optimized motion estimations in the next step are based on the optimized motion estimation in the adjacent previous ones. The new equations are

$$X(k \mid k-1) = X(k-1 \mid k-1), \tag{4.6}$$

$$P(k \mid k-1) = P(k-1 \mid k-1) + Q, \tag{4.7}$$

$$X(k \mid k) = X(k \mid k-1) + Kg(k)(Z(k) - X(k \mid k-1)), \tag{4.8}$$

$$Kg(k) = \frac{P(k \mid k-1)}{P(k \mid k-1) + R}, \tag{4.9}$$

$$P(k \mid k) = (1 - Kg(k))P(k \mid k-1). \tag{4.10}$$

The $X(1 \mid 1)$ is set to be -1, $P(1 \mid 1)$ is 1, Q is 1, R is 0.05. If the frames contain both local motion and global motion, there is just one difference between this circumstance and the one mentioned above. When the position being updated, the new position of the object's pixel will change based on both image correlation and optical flow results.

We do not add the Kalman Filter into the fixed positions, for example, at position $(1, 1)$ of a frame. We do not have the estimation data for this circumstance, so we can not update the estimation to get an optimized result. We could estimate state for the same pixel of the aim instead of the same position of the frames.

4.2 Spatial-Temporal Filter

The video sequences captured from the aircraft are quite noisy in nature. If we can get rid of some of the noise, the Kalman Filter will provide us with a better result. Based on the relationship of some neighbor pixels and the same pixel in several adjacent frames, we want to use the spatial or temporal filters before running Kalman Filter to delete the noisy data. The spatial filter is described as

$$P_{t_{new}} = \frac{1}{2}P_t + \frac{1}{16}(N_{t_1} + N_{t_2} + \cdots + N_{t_8}), \tag{4.11}$$

where P_t is the central pixel of a 3×3 block, N_{t_1} N_{t_2} $\cdots$ N_{t_8} are the eight neighbors of the central pixel P_t, $P_{t_{new}}$ is the new value of P.

The second filter is a temporal filter. In this approach the motion of a given pixel is correlated with motion of the same pixel in neighboring frames in time.

$$P_{t_{new}} = \frac{1}{2}P_t + \frac{1}{4}(P_{t-1} + P_{t+1}), \tag{4.12}$$

where P_t is the middle frame of three adjacent frames in a video sequence, P_{t-1} and P_{t+1} are the previous frame and the next frame, $P_{t_{new}}$ is the new value of P_t. In our experiment, we run the spatial filter first, save the filtered data, and then run the temporal filter on the saved data. We call this combination "spatial-temporal" filter. In our experience, the spatial-temporal filter works well.

In figs. 4.2, 4.3, 4.4, 4.5, and 4.6, the areas marked with circles are the local motions. All the figures here come from the same sequence as the previous figures. In fig. 4.3, the local motions are not obvious. In the global motion removal step, the image correlation can just get integral magnitude, so the global motion detected can not reflect the real sub-pixel displacement, there is error existing. Because of this reason, and also maybe caused by other factors, there is still noise after running image correlation and optical flow. In order to delete the disturbing noisy points, we set a threshold in the data pool. In the following figures, we know all the targets are moving upwards, so we delete all the other directions' information. After that, a clean result will be shown (fig. 4.4). After setting the threshold, we run the spatial filter first, we find that it could increase the density of the object's vector cluster (shown in fig. 4.5), since the filter constrains the relationship among the pixels come from the same region so it can help pixel modify its optical flow result according to the neighbors' data. Then, we run the temporal filter after the spatial filter (result is shown in fig. 4.6). The spatial-temporal filter increases the density of the targets further. It is good for the estimation process.

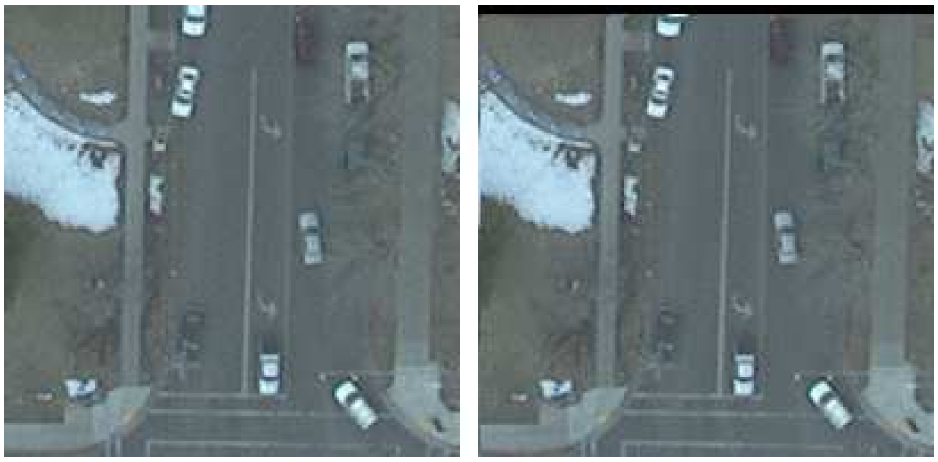


Fig. 4.2: One frame from aerial video (left) and its global motion removed version (right).

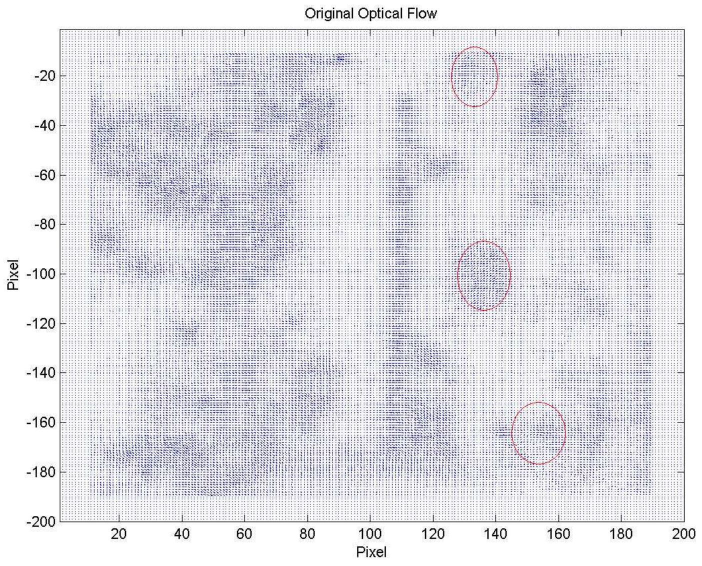


Fig. 4.3: The local motion detection.

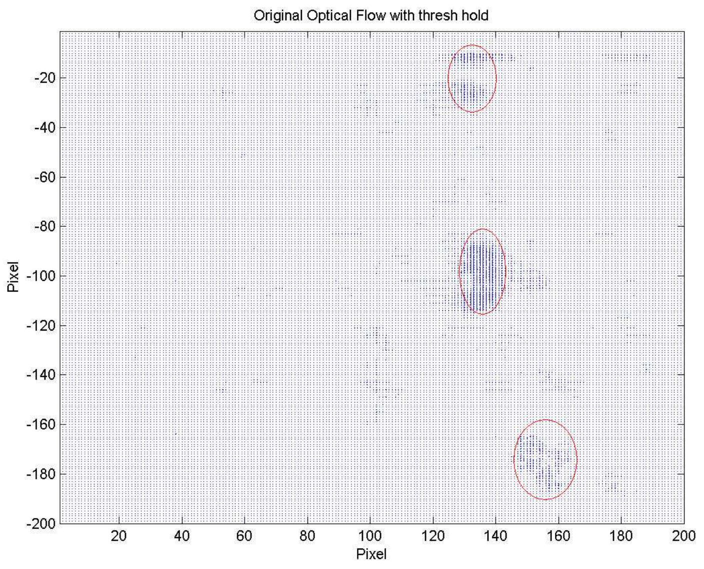


Fig. 4.4: The local motion detection with threshold.

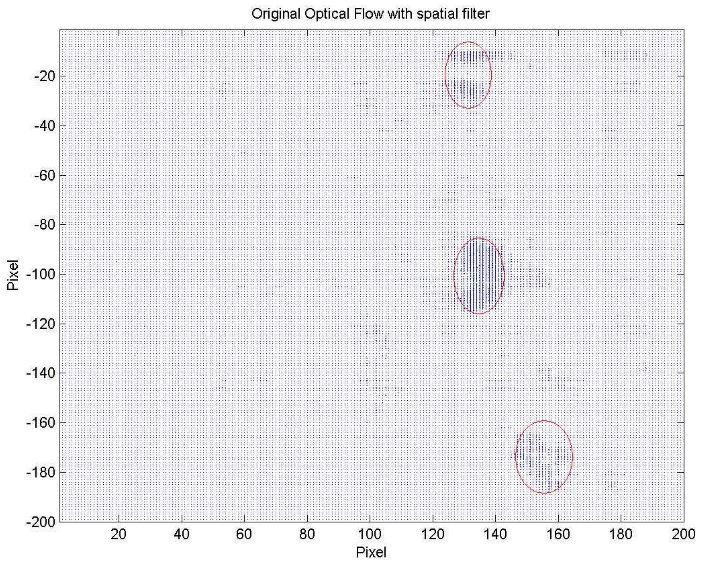


Fig. 4.5: The local motion detection with threshold and spatial filter.

4.3 Tracking Result

In fig. 4.7, the left picture comes from the first frame of an aerial video sequence, the middle one comes from the fifth frame of the same video, and the right one comes from the tenth frame. It shows the good tracking result. The Kalman Filter we used can track the object very well. (The code of the combination of threshold, spatial-temporal filter and Kalman Filter is shown in Appendix B.)

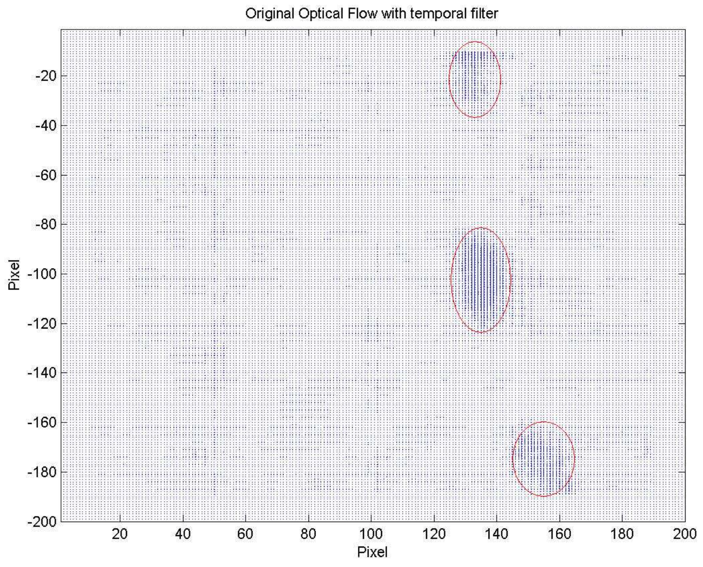


Fig. 4.6: The local motion detection with threshold and spatial-temporal filter.

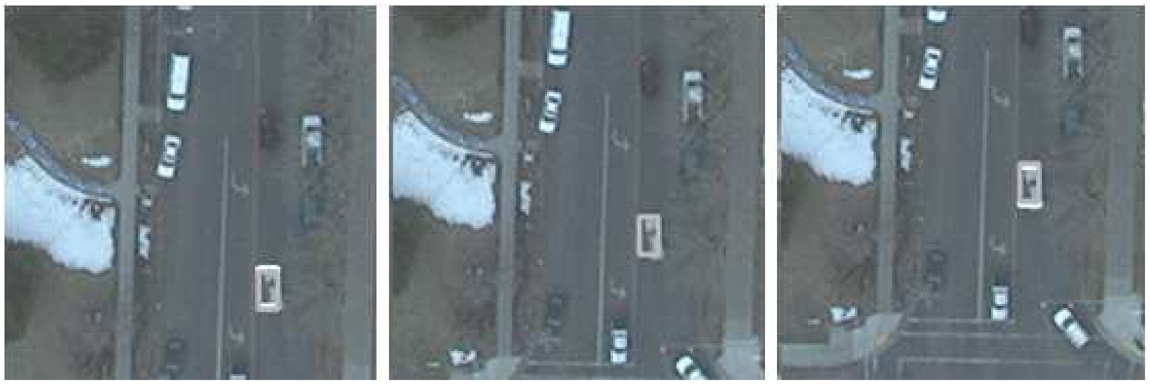


Fig. 4.7: The tracking result of the previous figure (the gray one is the Kalman Filter block and the white one is the optical flow block).

Table. 4.1 shows the tracking result. It reflects the motions in vertical direction of the objects in the frames. One is from Optical Flow data, and another one comes from the Kalman Filter estimation. In the first two frames, the Kalman Filter needs a period to converge. After the convergence, the Kalman Filter can estimate the motion well.

It is very difficult to avoid getting errors in the process of tracking. Errors may be caused by several factors. There is an explanation for why errors persist in the tracking result between each two adjacent frames. The following series of frames demonstrate the problem. Using the Optical Flow, not all frames were good enough to provide the perfect Optical Flow data. Figures. 4.8, 4.9, and 4.10 are the results of the combination of global motion removal, Optical Flow, threshold, and spatial-temporal filter. In some cases, we could get best case results which were robust to the noise (fig. 4.8). In other cases we have the tracking results containing a lot of noise (fig. 4.9). We may even get results that show no motion when motion really does exist (fig. 4.10). There should be a cluster of vectors of high density in the region of interest, but not all of them were shown there. Since it is difficult for us to completely eliminate global motion, as the image correlation errors existed in the frames. The imperfect objects and backgrounds also contribute to the errors. Due to these problems, it is difficult to always get perfect data for tracking. Finding sequences that have clean backgrounds and objects could help improve the tracking accuracy.

Table 4.1: Comparison between optical flow and Kalman Filter estimation for each frame (coefficients setting is explained in sec. 4.1).

Frame Number	*Optical Flow*	*Kalman Filter*
Frame1	-0.50642	-1
Frame2	-0.49838	-0.51061
Frame3	-0.55926	-0.55705
Frame4	-0.5661	-0.56569
Frame5	-0.66467	-0.66016
Frame6	-0.70376	-0.70177
Frame7	-0.43438	-0.44656
Frame8	-0.51412	-0.51104
Frame9	-0.46289	-0.46508
Frame10	-0.65887	-0.65004

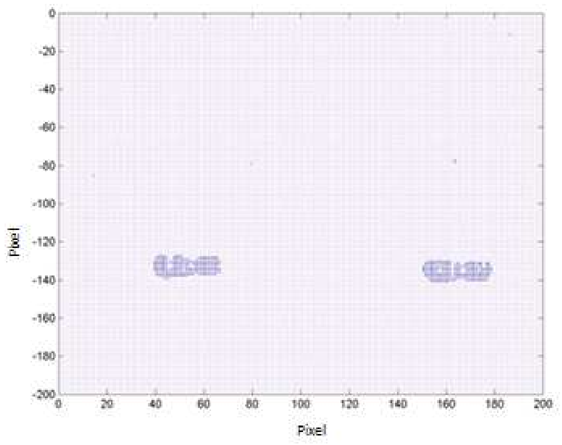


Fig. 4.8: Best case result.

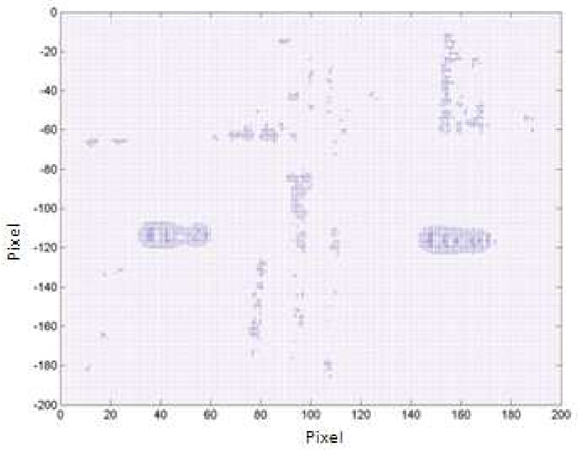


Fig. 4.9: Tracking result with noise.

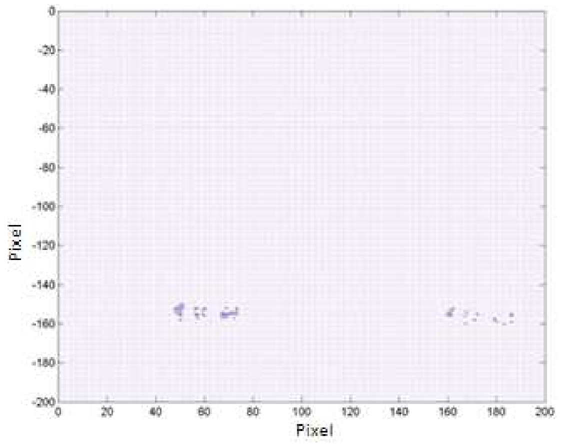


Fig. 4.10: Worst case result.

Chapter 5

Conclusion

From the experiments of our project, we were able to come to a conclusion. Optical flow shows good results to detect and track the local motion, but suffers from some problems. One problem is when there exists both local and global motion. If these two kinds of motion exist at the same time, we should use some method to remove the global motion first and then run optical flow to detect the remaining motion. In our project, we used the image correlation first to delete the global motion and then ran the optical flow to get the local motions. We also added the Kalman Filter to the project to smooth the motion history and estimate the future trajectory of objects. The threshold and spatial-temporal filter helped the Kalman Filter deleting most of the noise efficiently. From the experimental results we see that this idea could improve tracking results for aerial video. Errors exist in the tracking process. Some are caused by the quality of the video, and others may caused by the algorithms limitations. In the future work, we will talk about something to improve the tracking system performance.

Chapter 6

Future Work

The optical flow algorithms can work well when there is no illumination changes and when the objects' motions are small. In the real world, it is impossible to have a sequence without large illumination changes and it is difficult to constrain the objects' motions to be small. In order to solve these problems, some improvements in the algorithms are needed.

6.1 Pyramid Method

Coarse-to-fine strategy is a hierarchical process. It is used when large displacement exists in the image sequences. When we applied the coarse-to-fine strategy, we found that it is not good for small targets in aerial videos. Based on our experience, coarse-to-fine strategy is an efficient method if the targets are large.

6.2 Gray Value Constraints and Spatial-Temporal Smoothness Constant

Previously we just defined a gray value constancy. The old constancy assumption works fine in many cases, algorithms that rely only on image gray value constant constrain can not deal with image sequences with either local or global change in illumination. For image sequences where such cases appear, other constancy assumptions that are invariant against brightness changes can be applied. Invariance can be ensured, for instance, by considering (spatial) derivatives. We could also assume that the spatial gradients of an image sequence is constant during motion. Consider the higher order derivatives for the formulation of constancy assumptions. One of these is the Hessian matrix, another is the Laplacian constancy. Spatial-temporal smoothness approaches can also ameliorate the results simply by using the information of an additional timing dimension [26–29].

References

[1] R. Verri and T. Poggio, “Motion field and optical flow: qualitative properties,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 490–498, 1989.

[2] M. Tistarelli, F. Guarnotta, D. Rizzieri, and F. Tarocchi, “Application of optical flow for automated overtaking control,” pp. 105–112, Dec. 1994.

[3] K.-T. Song and J.-H. Huang, “Fast optical flow estimation and its application to real-time obstacle avoidance,” vol. 3, pp. 2891–2896, 2001.

[4] B. K. P. Horn and B. G. Schunck, “Determining optical flow: a retrospective,” *Artificial Intelligence*, vol. 59, no. 1–2, pp. 81–87, 1993. [Online]. Available: `http://www.sciencedirect.com/science/article/B6TYF-4811SSS-1Y/2/2ecf1379d9da8d5dc3b398e044b5a24e`.

[5] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” in *Proceedings: Imaging Understanding Workshop*, pp. 121–130, 1981.

[6] D. Li, *Restoration of Atmospheric Turbulence Degraded Video Using Kurtosis Minimization and Motion Compensation.* Ph.D. dissertation, Georgia Institute of Technology, Atlanta, 2007.

[7] D. H. Frakes, J. W. Monaco, and M. J. T. Smith, “Suppression of atmospheric turbulence in video using an adaptive control grid interpolation approach,” in *ICASSP 2001: Proceedings of the Acoustics, Speech, and Signal Processing on IEEE International Conference*, pp. 1881–1884. Washington, DC: IEEE Computer Society.

[8] J. P. Lewis, “Fast normalized cross-correlation,” in *Vision Interface*, pp. 120–123. Canadian Image Processing and Pattern Recognition Society, 1995. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.6062`.

[9] K. Briechle and U. D. Hanebeck, “Template matching using fast normalized cross correlation,” in *Proceedings of SPIE: Optical Pattern Recognition XII*, vol. 4387, pp. 95–102, Mar. 2001. [Online]. Available: `http://dx.doi.org/10.1117/12.421129`.

[10] M. Gelautz and D. Markovic, “Recognition of object contours from stereo images: an edge combination approach,” in *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 774–780, 2004.

[11] T. Lindeberg, “Edge detection and ridge detection with automatic scale selection,” in *CVPR: Proceedings of the Conference on Computer Vision and Pattern Recognition*, p. 465. Washington, DC: IEEE Computer Society, 1996.

[12] A. Bruhn, J. Weickert, and C. Schnörr, “Combining the advantages of local and global optical flow methods,” in *Pattern Recognition, L. Van Gool*, pp. 454–462. Verlag Berlin Heidelberg: Springer, 2002.

[13] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International Journal of Computer Vision*, vol. V61, no. 3, pp. 211–231, Feb. 2005. [Online]. Available: `http://dx.doi.org/10.1023/B:VISI.0000045324.43199.43`.

[14] T. Amiaz, E. Lubetzky, and N. Kiryati, "Coarse to over-fine optical flow estimation," *Pattern Recognition*, vol. 40, no. 9, pp. 2496–2503, 2007.

[15] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Transactions on Communications*, vol. COM-31,4, pp. 532–540, 1983. [Online]. Available: `citeseer.ist.psu.edu/burt83laplacian.html`.

[16] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.5313`.

[17] D. Gibson and M. Spann, "Robust optical flow estimation based on a sparse motion trajectory set," *Image Processing, IEEE Transactions*, vol. 12, no. 4, pp. 431–445, Apr. 2003.

[18] S. Lim and A. El Gamal, "Optical flow estimation using high frame rate sequences," in *Image Processing, International Conference*, vol. 2, pp. 925–928, Oct. 2001.

[19] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, pp. 433–467, 1995.

[20] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden, "Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[21] S. Jamkar, S. Belhe, S. Dravid, and M. S. Sutaone, "A comparison of block-matching search algorithms in motion estimation," in *ICCC: Proceedings of the 15th international conference on Computer communication*, pp. 730–739. Washington, DC: International Council for Computer Communication, 2002.

[22] N. Ohnishi and A. Imiya, "Dominant plane detection from optical flow for robot navigation," *Pattern Recognition Letters*, vol. 27, no. 9, pp. 1009–1021, 2006.

[23] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computer Surveys*, vol. 38, no. 4, p. 13, 2006. [Online]. Available: `http://dx.doi.org/10.1145/1177352.1177355`.

[24] D. Douglas and M. Dimitris, "Optical flow constraints on deformable models with applications to face tracking," *International Journal of Computer Vision*, vol. 38, no. 2, pp. 99–127, 2000.

[25] G. Welch and G. Bishop, "An introduction to the kalman filter." [Online]. Available: `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`.

[26] X. Ren, "Local grouping for optical flow," in *Computer Vision and Pattern Recognition. IEEE Conference*, pp. 1–8, Jun. 2008.

[27] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *International Journal of Computer Vision*, vol. 67, no. 2, pp. 141–158, Apr. 2006.

[28] J. Diaz, E. Ros, F. Pelayo, E. Ortigosa, and S. Mota, "Fpga-based real-time optical-flow system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 2, pp. 274–279, Feb. 2006.

[29] J. Barron and M. Khurana, "Determining optical flow for large motions using parametric models in a hierarchical framework," in *Proceedings of Vision Interface (V197)*, pp. 47–56, Kelowna, B.C., May, 1994.

Appendices

Appendix A

Optical Flow Algorithms

```
//optical flow using opencv lib's avy
#include <iostream>
#include <fstream>
#include "cv.h"
#include "cxcore.h"
#include "highgui.h"

int main()
{
using namespace std;

for(int index=1;index<=1;index++)
{
cout<<"now work for the "<<index<<" frame:"<<endl;
    char imgname1[100];
char imgname2[100];
char filename3[100];
char filename4[100];

int a=sprintf(imgname1,".../rect%d.bmp",index);
int b=sprintf(imgname2,".../rect%d.bmp",index+1);
int s=sprintf(filename3,".../ofx%d.txt",index);
int t=sprintf(filename4,".../ofy%d.txt",index);
```

```
ofstream outfilex;
outfilex.open(filename3);
ofstream outfiley;
outfiley.open(filename4);

IplImage* img1=cvLoadImage(imgname1,1);
if(img1==NULL)
{
  cout<<"image 1 is not loaded\n";
  return 0;
}
IplImage* img2=cvLoadImage(imgname2,1);
if(img2==NULL)
{
  cout<<"image 2 is not loaded\n";
  return 0;
}

//change rgb to gray
IplImage* img1g=cvCreateImage(cvSize(img1->width,img1->height), IPL_DEPTH_8U,1);
IplImage* img2g=cvCreateImage(cvSize(img1->width,img1->height), IPL_DEPTH_8U,1);
cvCvtColor(img1,img1g,CV_RGB2GRAY);
cvCvtColor(img2,img2g,CV_RGB2GRAY);

//optical flow

CvSize winSize;
```

```
winSize.height=15;

winSize.width=15;



IplImage* testVelocityX=cvCreateImage(cvSize(img1->width,img1->height),

IPL_DEPTH_32F,1);

IplImage* testVelocityY=cvCreateImage(cvSize(img1->width,img1->height),

IPL_DEPTH_32F,1);

cvCalcOpticalFlowLK(img1g,img2g,winSize,testVelocityX,testVelocityY);

//cvCalcOpticalFlowHS(img1g,img2g,1,testVelocityX,testVelocityY,0.001,

cvTermCriteria(CV_TERMCRIT_EPS+CV_TERMCRIT_ITER,60,0.01));

/*IplImage* testVelocityX=cvCreateImage(cvSize(150,150),IPL_DEPTH_32F,1);

IplImage* testVelocityY=cvCreateImage(cvSize(150,150),IPL_DEPTH_32F,1);

cvCalcOpticalFlowBM(img1g,img2g,cvSize(2,2),cvSize(1,1),cvSize(4,4),0,

testVelocityX,testVelocityY);

*/

CvScalar xc,yc;

for (int y=0;y<testVelocityX->height;y++)

{

for(int x=0;x<testVelocityX->width;x++)

{

xc=cvGetAt(testVelocityX,y,x);

outfilex<<xc.val[0]<<endl;

yc=cvGetAt(testVelocityY,y,x);

        outfiley<<yc.val[0]<<endl;

}

}
```

```
cvReleaseImage(&img1);
cvReleaseImage(&img1g);
cvReleaseImage(&img2);
cvReleaseImage(&img2g);
cvReleaseImage(&testVelocityX);
cvReleaseImage(&testVelocityY);



outfilex.close();
outfiley.close();
}

return 0;
}
```

Appendix B The Code of the Combination of Threshold, Spatial-Temporal Filter, and Kalman Filter

```
%% read data from .txt file and show vectors
clc,clear;
close all;

%% read from the .txt file
filename1='ofx%d.txt';
filename2='ofy%d.txt';
filename3='spatialofx%d.txt';
filename4='spatialofy%d.txt';
filename5='temporalofx%d.txt';
filename6='temporalofy%d.txt';
filename9='zx.txt';
filename10='zy.txt';
imgname='original%d.jpg';
imgname1='original_shreshhold%d.jpg';
imgname2='original_spatial_filter%d.jpg';
imgname3='original_temporal_filter%d.jpg';
for index=2:30
    a=sprintf(filename1,index);
    b=sprintf(filename2,index);
    fid=fopen(a);
    [flowx,countx]=fscanf(fid,'%f');
    fclose(fid);
    fid=fopen(b);
    [flowy,county]=fscanf(fid,'%f');
    fclose(fid);

%%change the data to the image format
k=1;
while k<40001
    for i=1:200
        for j=1:200
            otx(i,j)=flowx(k);
            sh_otx(i,j)=flowx(k);
            oty(i,j)=flowy(k);
            sh_oty(i,j)=flowy(k);
            k=k+1;
            if sh_oty(i,j)>0
                sh_oty(i,j)=0;
            end
            if sh_otx(i,j)>0
                sh_otx(i,j)=0;
            end
        end
    end
end

%%delete edge effect
otx(1:200,1:10)=0;
otx(1:200,190:200)=0;
otx(1:10,1:200)=0;
otx(190:200,1:200)=0;
oty(1:200,1:10)=0;
oty(1:200,190:200)=0;
oty(1:10,1:200)=0;
oty(190:200,1:200)=0;

sh_otx(1:200,1:10)=0;
sh_otx(1:200,190:200)=0;
sh_otx(1:10,1:200)=0;
sh_otx(190:200,1:200)=0;
sh_oty(1:200,1:10)=0;
sh_oty(1:200,190:200)=0;
sh_oty(1:10,1:200)=0;
sh_oty(190:200,1:200)=0;

%%check the arrangment
% for i=30:60
%     for j=40:120
%         otx(i,j)=0;
%     end
% end
%%make quiver
figure(1);
[x,y]=meshgrid(1:200,1:200);
h(index)=quiver(x,-y,otx,-oty);
axis tight;
title('Original Optical Flow');
```

```
xlabel('Pixel');
ylabel('Pixel');
c=sprintf(imgname,index);
saveas(h(index),c,'jpg');

sh_otx(1:200,1:200)=0;
figure(2);
[x1,y1]=meshgrid(1:200,1:200);
h1(index)=quiver(x1,-y1,sh_otx,-sh_oty);
axis tight;
title('Original Optical Flow with thresh hold');
xlabel('Pixel');
ylabel('Pixel');
c=sprintf(imgname1,index);
saveas(h1(index),c,'jpg');



%% finish making vectors


%% using colormap
% for i=1:200
%     for j=1:200
%         disotx(i,j)=abs(otx(i,j));
%     end
% end
% figure;
% imshow(disotx);
% title('disotx');
% colormap(hsv);
% caxis([-4 4]);
% colorbar('horiz');
% axis tight;
% title('Original Optical Flow in horizontal');
% xlabel('Pixel');
% ylabel('Pixel');
% for i=1:200
%     for j=1:200
%         disoty(i,j)=abs(oty(i,j));
%     end
% end
% figure;
% imshow(disoty);
% title('disoty');
% colormap(hsv);
% caxis([-4 4]);
% colorbar('horiz');
% axis tight;
% title('Original Optical Flow in vertical');
% xlabel('Pixel');
% ylabel('Pixel');
%% finish using colormap

%% spatial filter



pxnew(1:200,1:10)=0;
pxnew(1:200,190:200)=0;

pxnew(1:10,1:200)=0;

pxnew(190:200,1:200)=0;

pynew(1:200,1:10)=0;

pynew(1:200,190:200)=0;

pynew(1:10,1:200)=0;

pynew(190:200,1:200)=0;


for sfi=2:199

    for sfj=2:199

        pxnew(sfi,sfj)=1/2*sh_otx(sfi,sfj)+1/16*(sh_otx(sfi-1,sfj-1)+sh_otx(sfi-1,sfj)+s

        pynew(sfi,sfj)=1/2*sh_oty(sfi,sfj)+1/16*(sh_oty(sfi-1,sfj-1)+sh_oty(sfi-1,sfj)+s

    end

end

pxnew(1:200,1:200)=0;

figure(3);

[x2,y2]=meshgrid(1:200,1:200);

h2(index)=quiver(x,-y,pxnew,-pynew);

axis tight;

title('Original Optical Flow with spatial filter');

xlabel('Pixel');

ylabel('Pixel');

c=sprintf(imgname2,index);

saveas(h2(index),c,'jpg');


 % write the data into .txt files

    c=sprintf(filename3,index);

    fid=fopen(c,'w');
    dlmwrite(c,pxnew,'delimiter',' ','newline','PC');
    fclose(fid);
    d=sprintf(filename4,index);
    fid=fopen(d,'w');
    dlmwrite(d,pynew,'delimiter',' ','newline','PC');
    fclose(fid);
%% finish spatial filter

%%
end



% temporal filter
for index=3:29
    a=sprintf(filename3,index-1);
    b=sprintf(filename4,index-1);
    fid=fopen(a);
    [flowx1,countx]=fscanf(fid,'%f');
    fclose(fid);
    fid=fopen(b);
    [flowy1,county]=fscanf(fid,'%f');
    fclose(fid);

    c=sprintf(filename3,index);
    d=sprintf(filename4,index);
    fid=fopen(c);
    [flowx2,countx]=fscanf(fid,'%f');
    fclose(fid);
    fid=fopen(d);
    [flowy2,county]=fscanf(fid,'%f');
    fclose(fid);

    e=sprintf(filename3,index+1);
    f=sprintf(filename4,index+1);
    fid=fopen(e);
    [flowx3,countx]=fscanf(fid,'%f');
    fclose(fid);
    fid=fopen(f);
    [flowy3,county]=fscanf(fid,'%f');
    fclose(fid);

%%change the data to the image format
k=1;
while k<40001
    for i=1:200
        for j=1:200
            otx1(i,j)=flowx1(k);
            oty1(i,j)=flowy1(k);
            otx2(i,j)=flowx2(k);
            oty2(i,j)=flowy2(k);
            otx3(i,j)=flowx3(k);
            oty3(i,j)=flowy3(k);
            k=k+1;
        end
    end
end
```

```
%%temporal filter
for i=1:200
    for j=1:200
        newotx(i,j)=1/2*otx2(i,j)+1/4*(otx1(i,j)+otx3(i,j));
        newoty(i,j)=1/2*oty2(i,j)+1/4*(oty1(i,j)+oty3(i,j));
    end
end

newotx(1:200,1:10)=0;
newotx(1:200,190:200)=0;
newotx(1:10,1:200)=0;
newotx(190:200,1:200)=0;
newoty(1:200,1:10)=0;
newoty(1:200,190:200)=0;
newoty(1:10,1:200)=0;
newoty(190:200,1:200)=0;

 % write the data into .txt files
    y=sprintf(filename5,index);
    fid=fopen(y,'w');
    dlmwrite(y,newotx,'delimiter',' ','newline','PC');
    %fclose(fid);
    z=sprintf(filename6,index);
    fid=fopen(z,'w');
    dlmwrite(z,newoty,'delimiter',' ','newline','PC');
    %fclose(fid);
```

```
figure(4);
[x3,y3]=meshgrid(1:200,1:200);
h3(index)=quiver(x3,-y3,newotx,-newoty);
axis tight;
title('Original Optical Flow with temporal filter');
xlabel('Pixel');
ylabel('Pixel');
c=sprintf(imgname3,index);
saveas(h3(index),c,'jpg');

end

%% kalman filter data prepare part
row=168;
col=145;

fid=fopen('icx.txt','r');
[icx,countx]=fscanf(fid,'%f');
fclose(fid);
fid=fopen('icy.txt','r');
[icy,county]=fscanf(fid,'%f');
fclose(fid);

ofx_kal(1:20)=0;
ofy_kal(1:20)=0;

 %position update based on of
 for imgindex=3:22
```

```
     display(imgindex);

     a2=sprintf(filename5,imgindex);

     b2=sprintf(filename6,imgindex);

     fid=fopen(a2,'r');

     [ofx,count]=fscanf(fid,'%f',[200,200]);

     ofx=ofx';

     fclose(fid);


    fid=fopen(b2,'r');

    [ofy,count]=fscanf(fid,'%f',[200,200]);

    ofy=ofy';

    fclose(fid);


    ofx_update(imgindex-2)=ofx(row,col);

    ofy_update(imgindex-2)=ofy(row,col);


    row=row+ofy_update(imgindex-2)+icy(imgindex);

    col=col+icx(imgindex);


    row=round(row);

    col=round(col);


    ofx_kal(imgindex-2)=ofx_update(imgindex-2);

    ofy_kal(imgindex-2)=ofy_update(imgindex-2);


 end

%% kalman filter data prepare part finished
%% kalman filter main part
%ZX,ZY are the testing data
%SysX,SysY are the optimized state
%PX,PY are the optimized state covariance
%eSysX,eSysY are the estimated state
%ePX,ePY are estimated state covariance
%KgX,KgY are the kalman gain
filename7='ofx_kal.txt';
filename8='ofy_kal.txt';
ZX=ofx_kal;
ZY=ofy_kal;
SysX_final(1:20)=0;
SysY_final(1:20)=0;
PX(1)=1;
PY(1)=1;
SysX_final(1)=-1;
SysY_final(1)=-1;
%for index_outside=2:11
    index_outside=2;
    SysX(1:20)=-1;
    SysY(1:20)=-1;
    for index=index_outside:index_outside+8
    eSysX(index)=SysX(index-1);
    eSysY(index)=SysY(index-1);
    ePX(index)=PX(index-1)+1;
    ePY(index)=PY(index-1)+1;
    KgX(index)=ePX(index)/(ePX(index)+0.05);
    KgY(index)=ePY(index)/(ePY(index)+0.05);
    PX(index)=(1-KgX(index))*ePX(index);

    PY(index)=(1-KgY(index))*ePY(index);

    SysX(index)=eSysX(index)+KgX(index)*(ZX(index)-eSysX(index));

    SysY(index)=eSysY(index)+KgY(index)*(ZY(index)-eSysY(index));

    SysX_final(index)=SysX(index);

    SysY_final(index)=SysY(index);

    end

%end


fid=fopen(filename7);

dlmwrite(filename7,SysX_final,'delimiter',' ','newline','PC');

%fclose(fid);

fid=fopen(filename8);

dlmwrite(filename8,SysY_final,'delimiter',' ','newline','PC');

%fclose(fid);

fid=fopen(filename9);

dlmwrite(filename9,ZX,'delimiter',' ','newline','PC');

fid=fopen(filename10);

dlmwrite(filename10,ZY,'delimiter',' ','newline','PC');

fclose('all');
```