Algorithm for Object Recognition and Tracking on FPGA

Rosen Spirov[1], Dimiter Kovachev[2]

Abstract – **The paper presents the FPGA implementation for detecting the position of a moving object in a video sequence. By optical flow algorithms is possible to determine the approximate relative motion of the object. The frames are collected, subtracted from the background, filtered, enhanced and then the points are computed.**

Keywords– **Image processing, Adaptive filtering, Kalman filtering, FPGA, VHDL.**

I. Introduction

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 [1]. When the camera moves, a global motion will be added tothe local motion, which complicates the issue. Among the several tracking methods, point tracking is easy as shown in Fig.1.

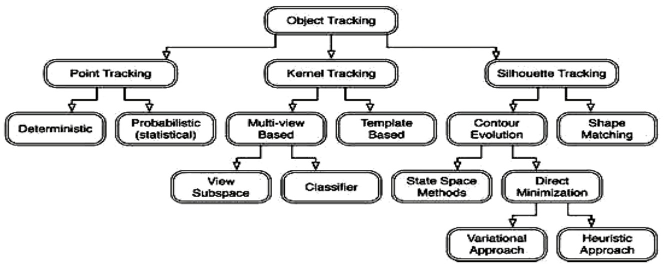

Fig.1The tracking methods

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 [2]. Having the data for all the pixels in each frame ensures that we can check each pixel's motion in each frame.

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 have to smooth the motion history, but have to estimate the motion into the next frame. The addition of a spatial-temporal filter improves the results in our later process.

[1]Rosen Spirov is with the Faculty of Electronic Engineering, TechnicalUniversity - Varna, 1, Studentska Str., 9010 Varna, Bulgaria,E-mail: rosexel@abv.bg.

[2]DimiterKovachev is with the Faculty of Electronic Engineering, TechnicalUniversity - Varna, 1, Studentska Str., 9010 Varna, Bulgaria, E-mail: dmk@abv.bg.

The basic hardware architecture, using extended Kalman filter-based method for calculating a trajectory by tracking features at an unknown location on Earth's surface, provided the topography is known, is given in [3]. The proposed model is implemented using VHDL and simulated and synthesized into an FPGA. The hardware design was implemented on an Altera DE2 board and Quartus II tools.

Traditionally Kalman filtering has proved to be satisfactorily in resolving many problems involved in predicting the position of moving targets [4], and is even useful for complex motion prediction. The capability of Kalman filtering to predict position allows us to overcome the artefact produced by this inherent processing latency, thus increasing the system'sreliable detection distance.

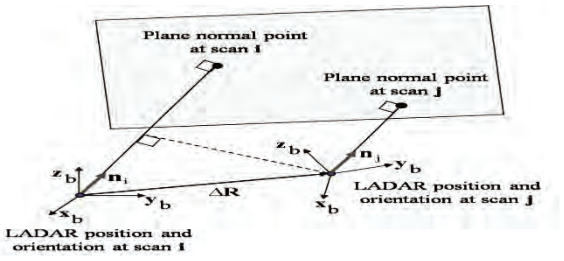
Plane normal point at scan i
Plane normal point at scan j
z_b
n_i
y_b
x_b
ΔR
z_b
n_j
y_b
x_b
LADAR position and orientation at scan j
LADAR position and orientation at scan i

Fig.2. Navigation-changes in perceived location of the normal point between scan i and scan j are applied to estimate position changes

In Fig.2ΔR is the delta position vector, displacement vector between scans *i*, at time *ti*, and scan *j*, at time *tj*, in this case); *ni* is the plane normal vector whose components are resolved in the Ladar body frame at scan epoch *ti; nj* is the plane normal vector whose components are. Note that in the navigation frame, the planar surface normal vectors at epoch's *ti* and *tj* are equal since resolved in the Ladar body frame at scan epoch *tj*; and, *ρi* and *ρj* are the shortest distances from the Ladar to the plane at epochs *ti* and *tj*, respectively stationary planar surfaces are assumed. However, expressed in the Ladar body frame both normal vectors are likely to be unequal due to the body frame rotation between epoch's *ti* and *tj*. From the geometry presented in Fig.1, a relationship can be derived between the projection of the displacement vector (between epoch's *ti* and *tj*) onto the planar surface normal vector and the change in the normal point range between scans *i* and *j* is shown in Eq.1:

$$\mathbf{\Delta R} \cdot \mathbf{n}_i = \rho_i - \rho_j \qquad (1)$$

Given M associated planar surfaces, a set of linear equations like (7) can be set up in matrix form is given in Eq.2:

$$\mathbf{H} \cdot \mathbf{\Delta R} = \Delta \rho \quad (2)$$

Were:

$$\mathbf{H} = \begin{bmatrix} \mathbf{n}_{i,1}^T \\ \vdots \\ \mathbf{n}_{i,M}^T \end{bmatrix}, \quad \mathbf{\Delta r} = \begin{bmatrix} \rho_{i,1} - \rho_{j,1} \\ \vdots \\ \rho_{i,M} - \rho_{j,M} \end{bmatrix} (3)$$

Note that a minimum of three non-collinear planar surfaces is required for the observation matrix, **H,** to be non-singular and thus allowing for a unique solution of Eq.2. The dynamic-state INS (Integrated Navigation Systems) calibration uses a Kalman filter to periodically estimate inertial error states. The estimation process is based on a complementary Kalman filter methodology [4] which employs differences between INS and laser scanner observables as filter measurements. Correspondingly, laser scanner observables of the Kalman filter are formulated as follows for the scan at time epoch t_m in Eq.4:

$$\Delta\rho_{LS}(t_m) = \begin{bmatrix} \rho_1(t_{m-1}) - \rho_1(t_m) \\ \ldots \\ \rho_N(t_{m-1}) - \rho_N(t_m) \end{bmatrix} \quad (4)$$

where N is the number of features for which is match is found time epoch t_m and t_{m-1}. Equivalent observables can be synthesized from INS measurements by transformation of the INS displacement vector into the range domain as follows in Eq.5 and Eq.6:

$$\Delta\rho_{INS}(t_m) = \mathbf{H}(t_{m-1})\left(\Delta\mathbf{R}_{INS}(t_m) + \Delta\mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_m)\mathbf{l}_{\mathrm{b}}\right) \quad (5)$$

$$\Delta\mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_m) = \mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_m) - \mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_{m-1}) \quad (6)$$

As mentioned previously, filter measurements are defined as differences between inertial and laser scanner observables Eq.7:

$$\mathbf{y}_{Kalman}(t_m) = \Delta\rho_{INS}(t_m) - \Delta\rho_{LS}(t_m) \quad (7)$$

The filter operates with dynamic states only. Particular filter states include: errors in position changes between consecutive scans, velocity errors, attitude errors, gyro biases, and accelerometer biases in Eq.8:

$$\delta\mathbf{x} = \begin{bmatrix} \delta\Delta\mathbf{R}_n^T & \delta\mathbf{v}_n^T & \psi^T & \mathbf{a}_b^T & \mathbf{b}_b^T \end{bmatrix}^T \quad (8)$$

For this state vector, the observation matrix $\mathbf{H}_{\mathrm{Kalman}}$ can be derived directly by augmenting the geometry matrix of Eq.3 with zero elements is shown in Eq.9:

$$\mathbf{H}_{Kalman}(t_m) = \begin{bmatrix} \mathbf{n}_1^T(t_{m-1}) & 0 & \mathrm{L} & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \mathbf{n}_M^T(t_{m-1}) & 0 & \mathrm{L} & 0 \end{bmatrix} \quad (9)$$

The measurement noise matrix $\mathbf{R}_{Kalman}$ is derived from the line and planar surface estimation processes performing a comprehensive covariance analysis of the feature extraction method. As a result, the current position error contributes to the position error for the next scan where the new line is used for navigation. A position drift is thus created. Statistics of image sequences and noise can be estimated if these signals are really stationary. Generaladaptive3-D spatial-temporal filters are very complex and prohibitive for real-time implementation[5]. One advantages of KalmanFilter is that it does not need all the motion history to estimate the next stage motion, itjust needs the nearest one from the current motion.

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 Eq.10:

$$Pt_{new} = 1/2Pt + 1/16(Nt_1 + Nt_2 + \ldots\ldots + Nt_8) \quad (10)$$

where **Pt**is the central pixel of a 3 x3 block, **$Nt_1 Nt_2 \ldots Nt_8$**are the eight neighbors of the central pixel **Pt**, **Pt*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 in Eq.11.

$$Pt_{new} = 1/2Pt + 1/4(Pt_{-1} + Pt_{+1}) \quad (11)$$

where Pt is the middle frame of three adjacent frames in a video sequence, Pt-**1** and Pt+**1**are the previous frame and the next frame, Ptnew is the new value of Pt. In our experiment, we run the spatial filter first, save the filtered data, and then run the temporal filter on the saved data. This combination is a spatial-temporal filter. In our work, the spatial-temporal filter works well. The areas marked with circles are the local motions, shown in fig.3.

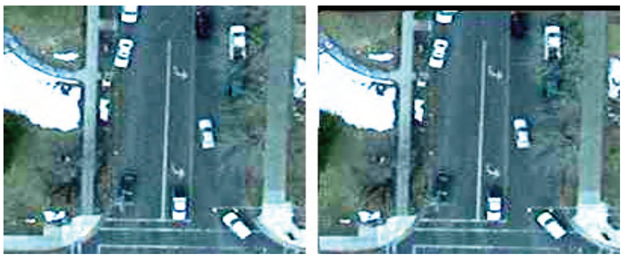

Fig.3The frame from aerial video and its global motion removed version

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. 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 in fig. 4.

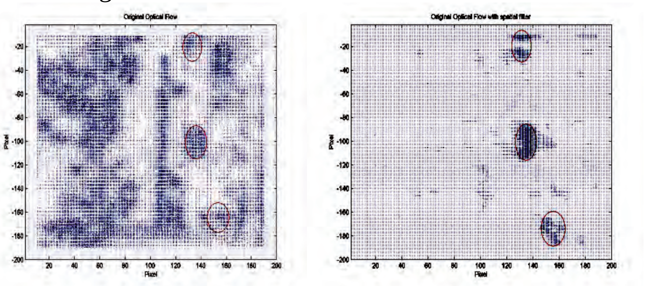

Fig.4 The local motion detection and with threshold and spatial filter

After setting the threshold, we run the spatial filter first, we find that it could increase the density of the object's vector cluster, 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. The spatial-temporal filter increases the density of the targets further. It is good for the estimation process.

II.ALGORITHM FOR OBJECTRECOGNITION

Each frame is fed into the program which subjects the frame to process of object recognition to achieve a noise free enhanced image containing only the object. At the rate of 1 frame per second, the enhanced image is fed to the tracking program. This analyzes each frame and computes the first white pixel that represents the object. This is under the assumption that object is made up of uniform material. This point is then plotted on a new image as the first position of the object. Subsequent frames are collected, subtracted from the background, filtered, enhanced and then the points are computed. The program of the combination of Threshold, Spatial-Temporal Filter, and Kalman Filter acquires the frames and plots the individual points. The code is:

```
%% 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……
%% spatial filter………………
%%change the data to the image format…….
%%temporal filter…………
%% 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
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');
```

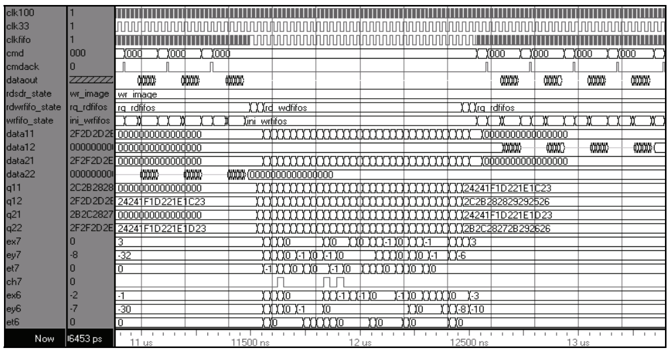

Fig.5 System simulation on ModelSim-Altera.

The advantage of parallel processing in FPGA leads to a substantial increase in performance and accuracy in processing, extraction of information than in the simulation in Matlab as shown in fig.6 and simulation in fig.7.

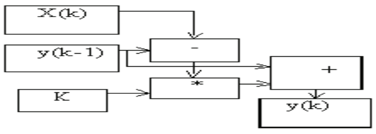

Fig.6 Pixel Kalman Filter Calculation

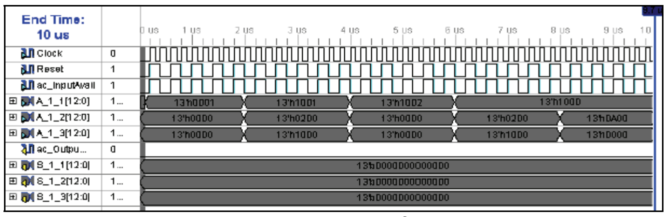

Fig.7 Behavioral simulation of the Kalman Filter

III. Results

The above hardware design was implemented on an Altera Quartus II board and ModelSim, shown in fig.8and was able to operation time is about 60 clock cycle, which about 0.6us at 100MHz clock pulse, so the operation speed can be up to 1.5MHz. The whole design requires 4168 ALUTs and 241 registers- occupancy of resources is about 49% as in Tab.1.

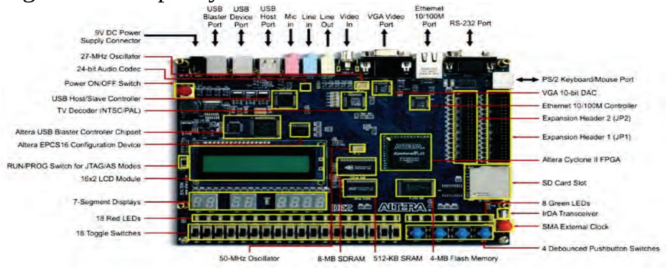

Fig.8 The Altera DE2 board

TABLE I

Sr. No.	Information	Count	%use
1	Noofslice	2145of32640	7%
2	Slice LUTs	3626of32640	14%
3	SliceLUTs Used as logic	3626of32640	14%
4	LUTFlip-Fl.pair used	4168	
5	LUT Flip-Fl. pairs with	2023of4168	49%
6	LUT Flip-Flop pair	542of4168	17%
7	Fully used FFpairs	1603of4168	41%
8	BondedIOBs	82 of 480	21%
9	DSP48Es	16 of 288	7%

The output and parameters are alignet such that one memory controller can handle reads and writes to input buffers. Hardware resources for the parameter calculation approximately 1664 Logic Cells. When we compare the outputs obtained from Matlab and FPGA, we find the outputs obtained using the AlteraDE2CycloneIIFPGAkitare computationally efficient. The pixel values are scaled and the outputs are comparable to the ones obtained using Matlab. Comparison between Optical flow [6] and Kalman Filter estimation shows the tracking result in Tab2.

TABLE II

Frame Number	Optical Flow	Kalman Filter
Frame1	-0.49883	-1
Frame2	-0.48995	-0.50219
Frame3	-0.48615	-0.54989
Frame4	-0.55982	-0.55775
Frame5	-0.65988	-0.65408
Frame6	-0.71012	-0.70244
Frame7	-0.42995	-0.43128
Frame8	-0.55981	-0.50974
Frame9	-0.45776	-0.46228
Frame10	-0.64996	-0.64356

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.

IV. 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. 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.

References

[1] Maybeck, Peter S. Stochastic Models Estimation and Control, Vol I. Academic Press, Inc., Orlando, 32887.

[2] Hagen, E. Navigation by Optical Flow, In Proc. of IAPR Intern.Conf.-Patt. Recgn. Vol.1, 1992, pp. 700–703.

[3] Lazarov, A. D. Spatial correlation algorithm for ISAR image Reconstruction- 2000 IEEE - Rad.Conf. Virginia, USA, 7-12 P.

[4] Brown and P. Y. C. Hwang, Introduction to Random Signals and Applied Kalman Filtering, 3 rd Ed.2006.

[5] Pellerin D.and S. Thibault. Practical FPGA programming in C.Prentice Hall PTR, ISBN: 0-13-154318-0.ANN

[6] J. Diaz, E. Ros,S. Mota,Fpgabased real-time optical flow system, IEEE Transactions on, vol. 16, pp.274-279, Feb. 2006.