

Vector Bank Based Target Tracking via Vision Sensors in Aviation Systems

Wei Zhao[†], Jeffrey Fan[†], Asad Davari[§]

[†]Department of Electrical and Computer Engineering, Florida International University, Miami, Florida

[§]Department of Electrical and Computer Engineering, West Virginia University Institute of Technology, Montgomery, West Virginia

Abstract—In this paper, we present a technique of detection and tracking of one or possibly more moving objects by implementing a "Vector Bank" into an H.264 based System-on-a-Chip design architecture as a vision sensor in aviation systems, such as Unmanned Aerial Vehicles (UAV). Most of the existing target-tracking algorithms are based on software solutions. The introduction of H.264 encoding chips provides a far better hardware solution by developing motion estimation blocks to detect Macro-Block movement up to 10 reference frames in streaming imaging data. By adding a vector bank with the proposed framework, a high-resolution H.264 based vision sensor could be easily developed and implemented for target tracking purpose in a real-time manner in aviation systems.

I. INTRODUCTION

The moving object detection and tracking have been studied for years. It has been widely used in application to computer vision, such as robotics, video surveillance, authentication systems and machine-human interfaces. In tracking an object or multiple objects in aviation system, such as Unmanned Aerial Vehicles (UAV), the response time for tracking becomes critical, as the control and command decision needs to be made in a real time manner. Before we present this paper, there are many existing approaches to track moving objects as you could find in [1]-[6]. Most of which use the frame differences of the image to detect moving objects and obtain the object boundary. These approaches, however, may be quite time-consuming [7]. In order to improve the computational efficiency, a hardware solution, such as, H.264 System-on-a-Chip (SoC) based framework, is introduced in this paper.

The popular video encoding and decoding standard, the so-called H.264/AVC [8], which is developed by the ITU-T (International Telecommunication Union) and MPEG (ISO/IEC Moving Picture Experts Group), also known as MPEG-4 Part 10, provides a much efficient way in compressing the video data. The new standard may provide about 50% in size reduction in comparison to the older standard, such as MPEG-2 [9].

Figure-1 is a general architecture for a common H.264 encoder core [10]. In this figure, the most important part is the Motion Estimation (ME) block. Basically, we detect an object in motion by the movement of the color from our eyes (as human sensors). It isolates the object from the background. The ME block uses well-defined algorithms to compress a significant ratio of video residue by detecting the movement of all the objects from the current frame [10].

In this paper, we propose a theoretical architecture and framework to build a vision sensor with the capabilities of

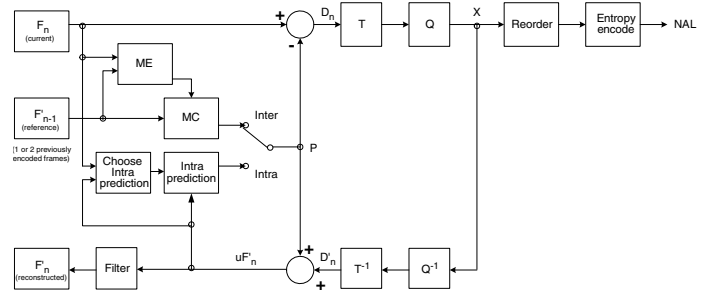


Fig. 1. General H.264 Encoder Core Architecture

detecting and tracking one or more rough-shaped objects in motion. Such a framework is based on the H.264 SoC codec module. With a programmable interface, the control and command center may choose to either follow the object directly or employ a more sophisticated algorithm to identify and locate the object accurately, such that a tactical decision can be made in a real time manner.

II. MOTION VECTOR BANK

A. Motion Vectors in H.264 Codec

Figure-2 shows a general view of "Motion Vectors" in some portion of a reference frame [10][11]. A Macro-Block (MB) with 16x16 square color pixels of the original frame represents a single block here. In fact, the H.264 ME may generate MB with motion vectors that are smaller than 16x16. They may be 16x8, 8x8, or even 4x4 square color pixels. In this paper, we assume those MBs are all 16x16 to simplify our design. We also assume the desired tracking is single object. However, the proposed techniques may be applied to track multiple objects. From the figure, we could easily tell the differences between background and the object, even though they may be very colorful. However, being colorful could add some challenges in edge detection of the object via color differences [11].

For every MB in H.264 algorithm, it uses differential coordinates (in both X, Y directions) to indicate the vector that the current MB is deviated from the most similar MB in the reference frame bank. We need these data for the location of the moving object. We also need them for reference by storing several frames of data somewhere in the memory in order to predict the movement in the future.

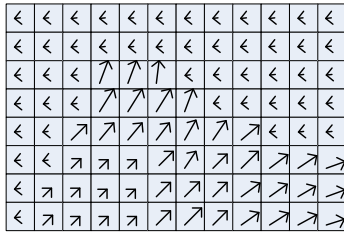


Fig. 2. Illustration of Motion Vectors generated from Motion Estimation

B. Motion Vector Bank

For any format of video frame, we assume that it contains M Macro-Blocks in the horizontal direction and N Macro-Blocks in the vertical direction. Then, the matrix of differential pixels becomes:

$$\begin{pmatrix} (X, Y)_{(0,0)} & (X, Y)_{(1,0)} & \dots & \dots & (X, Y)_{(M,0)} \\ (X, Y)_{(0,1)} & (X, Y)_{(1,1)} & \dots & \dots & (X, Y)_{(M,1)} \\ (X, Y)_{(0,2)} & (X, Y)_{(1,2)} & \dots & \dots & (X, Y)_{(M,2)} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ (X, Y)_{(0,N)} & (X, Y)_{(1,N)} & \dots & \dots & (X, Y)_{(M,N)} \end{pmatrix} \quad (1)$$

For every single "X" and "Y" here, they are accurate in pixels. They will require different size of space to be stored with different resolution of videos. For example, for typical High Definition (HD) 1080p video, we need 11 digits for "X" plus "Y", so that they could indicate from 0 to 1920 or 1080.

To build a "Motion Vector Bank", we will first need to know the size of the "Bank". Again, for a 1080p typical HD video, a single frame vector bank will need:

$$\left(\frac{1920}{16} \times 11\right) \times \left(\frac{1080}{16} \times 11\right) = 980,100 \text{ bits} \quad (2)$$

$$= 122,512.5 \text{ bytes}$$

The estimate cost for 128KB of cache memory is affordable, but if 5 frames of vector bank are what we need, it may be a problem. For the implementation of the multi-frame vector banks, there is a trade-off that we could do by adding certain control logic to save a lot of memory space. As you can see in the Figure- 3, the object we are tracking is normally inside the frame by less than 1/5 picture size. Furthermore, we don't need to locate and calculate those background vectors, since they are considered as "noise" and can be zeroed out in most cases. If we just record what we are interested, with the cost of storing 7 digits of coordinates, we could easily put 5 frames of vectors into 1 frame of the buffer. However, the control mechanism will be dynamic because maybe there are 10 frames of vectors or maybe there are only 3 depending upon the movement of the object.

In the proposed Vector Bank, it stores the vectors coming from the Motion Estimation Block of H.264 encoding core. The data inside the core indicates the moving directions of the object or objects. Since the tracking is vector-based, the Vector Bank can be used to track single object or multiple

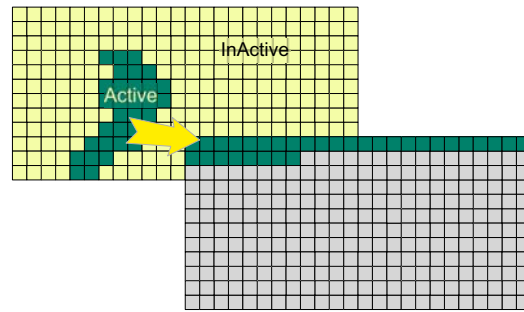


Fig. 3. Multi-frame vector data in one frame buffer with location indicator

objects by increasing the size of the memory buffer. The next step is to deal with the data we collect, and provide the feedback to the vision sensors to track the movement of the desired object or objects, and then send the object indicators to the command and control center.

III. MOTION OBJECT DETECTION AND SENSOR TRACKING ALGORITHM

It is straightforward for detecting objects in motion after we implement the Vector Bank. If there is any object coming inside of the frame, it should be already located inside the Vector Bank before it appears in the front screen of the observing monitor. If there are more than one active object were found, SoC should select either the balance weight point for the objects or the single object selected by some algorithms in the command and control center.

In other words, "Sensor Tracking" is a feedback system to adjust the angle for the sensor to track the moving object. It's limited by the moving condition of sensor. And the angle and many mechanical problems will affect the result of a tracking sensor too. Here, we just talk about those feedbacks that are based on the collected data.

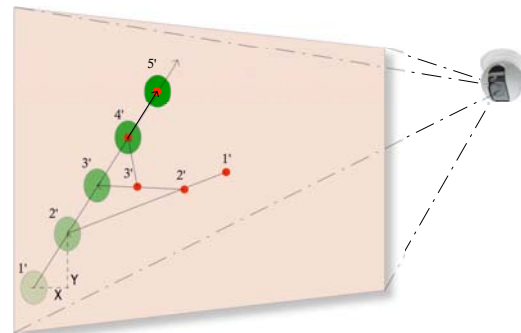


Fig. 4. Illustration of the tracking algorithm

As you can see from Figure-4, the algorithm for a tracking feedback system works like this:

- 1) The number here means time slot. Let's assume 1 second for the interval. So, 1', 2', 3', 4' and 5' indicate the time passing from the green spot to be engaged into our frame.

- 2) The fading green spot is our moving object. It comes into our frame and goes along the arrow line (from 1' to 5').
- 3) The red spot is the center point of our frame. It tracks the green spot's movement and tries to put the green spot right on top of it. The sensor receives the movement order from the feedback system and starts to move. This makes our red spot move too. For illustration purpose, we didn't draw several frames that could make our figure hard to understand.
- 4) The X and Y are the data coming from the Vector Bank. If we assume the time interval is based on per frame (1/30 or 1/60 sec per frame), then they are exactly the pixel data stored inside the vector bank. But for now, they are 30X and 30Y (assume X and Y never change).

The algorithm is quite straightforward: we could get the vector data from the beginning of the scenario at time 1'. It's easy for us to predict our object (green spot) will move to 2' by the next second. And, we simply move our center of frame to 2' till we get into the red spot 2'. At time instance 2', we do the same thing, as we've done in 1'. We predict the position of green spot by time instant 3', and we follow it until we arrive at red spot 3'. And we got red spot 4' and 5'.

- 1) During the time we move our red spot, we cannot analyze the vector for the green spot any longer, because the whole frame is moving. So, we have to refresh the Vector Bank after we get to a new time slot. It takes time for us to get the new vector and position.
- 2) If the vector of the green spot moves so fast that exceeds our maximum movement ability of the vision sensor, the object could not be tracked.
- 3) The movement under this scenario is a fixed number, it could be much more sophisticated movement in the real application, so that Digital Signal Processor (DSP) for movement tracking analysis may be needed due to this problem.

The actual math experiment is shown in Figure-5. Assume there is an HD sensor with a green center spot. It detects a moving object by our vector bank. And, now it needs to follow the vector bank's direction trying to make object more "center-like" in the next second. We assume that:

- 1) (x_i, y_i) is the position of the moving object at time i (blue track);
- 2) (x^C_i, y^C_i) is the position of the center spot of the frame at time i (green spot);
- 3) (x^E_i, y^E_i) is the position of the estimation of object movement at time i (red start spot);
- 4) L is the maximum distance that the vision sensor can be moved in a given unit time.
- 5) Assume that time unit is one second. We could get the vectors within the 1st frame of this very second. If

its 30 frame per seconds (fps) video, it means that we need $\frac{1}{30}$ seconds to obtain the vectors.

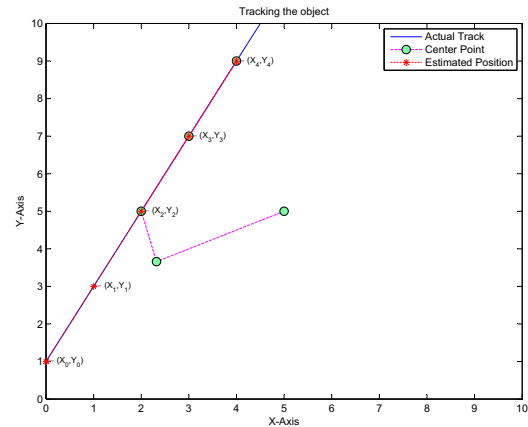


Fig. 5. Successful tracking of the object

So, for $i = 0, 1, 2, \dots, n$, record the current center point (x^C_i, y^C_i) of the vision sensor, and calculate the vector based on the first two frames:

$$V_x = 30 \times (x_{i+\frac{1}{30}} - x_i) \quad (3)$$

$$V_y = 30 \times (y_{i+\frac{1}{30}} - y_i) \quad (4)$$

Estimate the target point that the object is moving to:

$$x^E_{i+1} = x_i + V_x \quad (5)$$

$$y^E_{i+1} = y_i + V_y \quad (6)$$

Then, solve the linear equations to get k and b :

$$\begin{cases} kx^C_i + b = y^C_i \\ kx^E_{i+1} + b = y^E_{i+1} \end{cases} \quad (7)$$

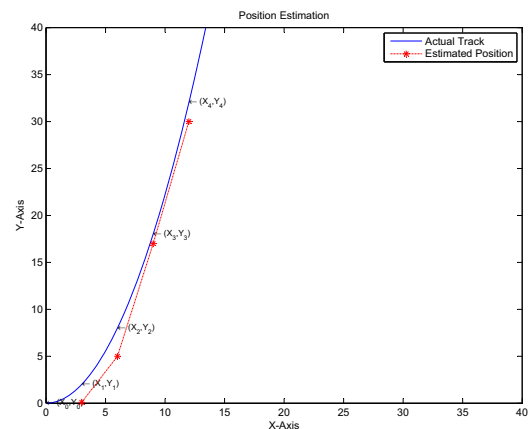


Fig. 6. Differences of tracking estimation

Calculate the distance l between the center point of the current frame (x^C_i, y^C_i) and the estimation point of next unit of time (x^E_{i+1}, y^E_{i+1}) :

$$l = \sqrt{(x^E_{i+1} - x^C_i)^2 + (y^E_{i+1} - y^C_i)^2} \quad (8)$$

If $l \leq L$, set:

$$(x_{i+1}^C, y_{i+1}^C) = (x_{i+1}^E, y_{i+1}^E) \quad (9)$$

else if $l > L$, set:

$$\begin{cases} x_{i+1}^C = x_i^C - (x_i^C - x_{i+1}^E) \frac{L}{l} \\ y_{i+1}^C = ky_{i+1}^E + b \end{cases} \quad (10)$$

Finally, the vision sensor obtains the feedback by moving to (x_{i+1}^C, y_{i+1}^C) . It is a successful tracking as shown in Figure-5. That's because the movement of the moving object is smaller in comparison to the observing window of the vision sensor. However, if the movement of the object is too big, then we cannot quite follow it. In reality, moving trace could be more sophisticated and hard to track. The estimation difference could be always there as shown in Figure-6. In the figure, the trace of the moving object is a x^2 curve. We have a linear moving estimation system to track nonlinear movement. The distance between the red spark spot and blue dot is the estimation difference. We can do a lot of things to reduce the estimation difference. For example, we may use a DSP to analyze the Motion Vector and give an accurate estimation after it goes through some motion calculation.

IV. MOTION OBJECT DETECTION WITH SENSOR TRACKING H.264 SOC ARCHITECTURE

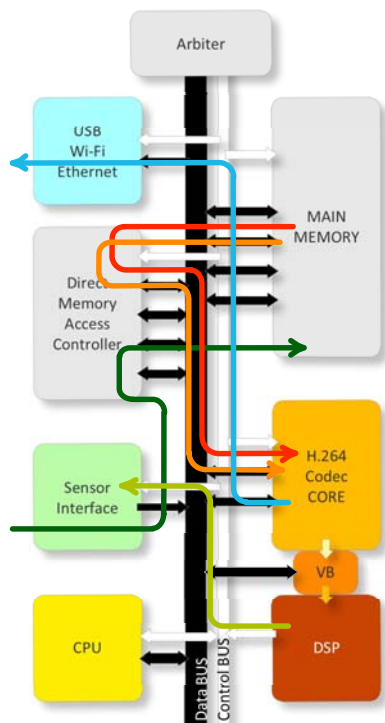


Fig. 7. Overall architecture of tracking system

Figure-7 shows the overall architecture for the system of Moving Object Detection and Tracking sensor chip system. The signal for the video sensor data goes into Sensor Interface (SI) along the path of Direct Memory Access Controller (DMAC) and gets into the Main Memory (MM)

shown as the dark green line in the figure. The red line and Orange line indicate the current frame and reference frame(s) going from MM along DMAC all the way to the H.264 codec core. System output is indicated in blue line that contains the compressed video data. It comes out from H.264 codec core and gets out of the system through any available data interface, such as USB/Wi-Fi/Ethernet or other any interfaces. You can find an orange Vector Bank (VB) that is attached under the H.264 Codec Core. It grabs the vector data from the Motion Estimation module of H.264 core, and organizes them. With an external DSP, we can do a lot of things with the data contained in VB, such as estimating the moving directions or analyzing the object type or moving targets. Finally, DSP sends out the data under analysis and converts them to the vision sensor, i.e. camera (green line) and the data interface connected to the main server. The CPU takes charge of the cooperation of every part of the system and the Arbiter insures that the utilization of the BUS system is at the maximum performance.

V. CONCLUSION

In this paper, we have proposed the concept of "Vector Bank" to be implemented in the standard H.264 platform. The framework can be used to track single or multiple objects up to the limitation of vision sensors. The control and command center may easily choose to either follow the object directly or employ a more sophisticated algorithm to identify and locate the object accurately, such that a tactical decision can be made in a real time manner in the modern aviation systems.

REFERENCES

- [1] H.Y. Zhang and B.Z. Yuan, "A multi-resolution image matching algorithm for moving object detection by wavelets", ISSIPNN Proceedings, vol. 1, pp. 276-279, April, 1994.
- [2] D. Li, "Moving objects detection by block comparison", Electronics, Circuits and Systems, vol. 1, pp. 341-344, Dec, 2000.
- [3] R. Cucchiara, C. Grana, M. Piccardi and A. Prati, "Statistic and knowledge-based moving object detection in traffic scenes", IEEE Proceedings. Intelligent Transportation Systems, pp. 27-32, Oct, 2000.
- [4] J.M. Odebez and P. Bouthemy, "Detection of multiple moving objects using multiscale MRF with camera motion compensation", ICIP 1994, vol. 2, pp. 257-261, Nov, 1994.
- [5] Y.K. Jung, K.W. Lee and Y.S. Ho, "Content-based event retrieval using semantic scene interpretation for automated traffic surveillance", IEEE Transactions on Intelligent Transportation Systems, vol. 2, pp. 151-163, Sep, 2001.
- [6] R. Montoliu and F. Pla, "Multiple parametric motion model estimation and segmentation", ICIP 2001, vol. 2, pp. 933-936, Oct, 2001.
- [7] T. Xia, C. Liu and H. Li, "An Efficient Moving Object Detection and Description", preprint 2005.
- [8] Joint Video Team of ITU-T and ISO/IEC JTC 1, Draft ITU-T recommendation and final draft international standard of joint video specification, Document JVT-GO50, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, 2003.
- [9] R. Chen, W. Zhao, Q. Liu, J. Fan, "Efficient H.264 architecture using modular bandwidth estimation", IEEE 5th International Conference on Embedded Software and Systems (ICCESS'08), pp. 277-282, Chengdu, China, July 29-31, 2008.
- [10] I. E. G. Richardson, "H.264 and mpeg-4 video compression", August 2003.
- [11] W. Zhao, Z. Luo, J. Fan, S. Tan, "Vector edge detection in H.264 Implementation", IEEE 5th International Conference on Embedded Software and Systems Symposia (ISHSO'08), pp. 208-212, Chengdu, China, July 29-31, 2008.