CHAPTER -4

ALGORITHM IMPLEMENTATION

4.1 Embedded IR Object Tracking Algorithms
4.2 Parallel Image Processing Algorithms
4.3 Implementation of SIMD Architectures
4.4 IR Image Pre-Processing Algorithms
4.5 Sum of Absolute Difference (SAD) Algorithm
4.6 Infra-red Image Segmentation
4.7 Radon Transform Projections
4.8 H.264 Based Moving Vectors
4.9 System on Chip Implementation
4.1 EMBEDDED IR OBJECT TRACKING

ALGORITHM: Video tracking

Computer vision algorithms are getting more and more important. The realization of such computer vision algorithms needs enormous computing power of the hardware. Cutting edge DSPs provide enough performance for computer vision algorithms if optimized algorithms are used. Highly optimized code with the use of intrinsics saves runtime and helps them to meet real time constraints. Porting the algorithm on an embedded system brings up questions like which hardware fits the needs of the application best, how fast can the algorithm be processed and how many resources are required. Selecting an embedded hardware has big influence on the performance.

Video tracking is the process of locating a moving object (or several ones) in time using a camera/FPA. An algorithm analyses the video frames and outputs the location of moving targets within the video frame. The main difficulty in video tracking is to associate target locations in consecutive video frames, especially when the objects are moving fast relative to the frame rate. Here, video tracking systems usually employ a motion model which describes how the image of the target might change for different possible motions of the object to track. BDTI is the most respected source for signal processing benchmarks. Our benchmarks are used by dozens of semiconductor vendors and thousands of chip users to evaluate, compare, and select signal processing engines. BDTI has benchmarked the signal processing performance of processors for nearly 15 years, and has expanded its benchmarking activities to include FPGAs, multi-core chips, and other technologies.
4.1.2 Simple motion models

The motion model is a 2D transformation (affine transformation or homography) of an image of the object (e.g. the initial frame). When the target is a rigid 3D object, the motion model defines its aspect depending on its 3D position and orientation. For video compression, key frames are divided into macroblocks. The motion model is a disruption of a key frame, where each macroblock is translated by a motion vector given by the motion parameters. The image of deformable objects can be covered with a mesh, the motion of the object is defined by the position of the nodes of the mesh. The role of the tracking algorithm is to analyse the video frames in order to estimate the motion parameters. These parameters characterize the location of the target.

Common algorithms: There are two major components of a visual tracking system:

1. Target Representation and Localization and
2. Filtering and Data Association.

Target Representation and Localization is mostly a bottom-up process. Typically the computational complexity for these algorithms is low. The following are some common

**Target Representation and Localization algorithms:** Blob tracking: Segmentation of object interior (for example blob detection, block-based correlation or optical flow). Kernel-based tracking (Mean-shift tracking): An iterative localization procedure based on the maximization of a similarity measure (Bhattacharyya coefficient). Contour tracking: Detection of object boundary (e.g. active contours or Condensation algorithm) Visual feature matching: Registration
Filtering and Data Association is mostly a top-down process, which involves incorporating prior information about the scene or object, dealing with object dynamics, and evaluation of different hypotheses. The computational complexity for these algorithms is usually much higher. The following are some common Filtering and Data Association algorithms: Kalman filter: An optimal recursive Bayesian filter for linear functions subjected to Gaussian noise. Particle filter: Useful for sampling the underlying state-space distribution of non-linear and non-Gaussian processes.

The Reconfigurable Architecture is domain-specific, not general-purpose. Therefore, right at the outset need to identify specific applications that will be targeted by the Reconfigurable Architecture. In experiment, choose H.264 for the target application, since it requires a lot of parallelized computation. Research on the architectures of parallel image processing systems is the basis for system designing using reconfigurable computers. It plays an important role in achieving optimal conversion from algorithm to structure so that it is helpful to design a highly efficient embedded system. This is a method for detecting and tracking moving objects which includes non-rigid objects. method is robust because use edge-based features which are insensitive to illumination changes. The method is also fast because the area of edge-based features is less than region based features.

In an infrared image processing real-time requirements are very important issues. In future seekers the object detection below 10 ms will be indispensable. This can only be met by over-sized DSP-/microcontroller working in a pixel serial manner with a high system clock. In this research work few parallel processor architecture are developed for FPGAs, which is based on optimized algorithms. It is applicable for
multiple object detection in seeker image processing with reconfigurable hardware. The processors are connected by means of an HDL language. They can fulfil multiple algorithms such as the simple erosion or the summation of pixels. This summation leads to the calculation of moments and allows the determination of the centroid as well as the rotation angle of objects. Contrary to a fully parallel architecture, the image here is processed piece by piece so that a FPGA is sufficient for the whole design. The greatest advantages of this approach are the short processing time, the small design size and the low clocking. Nevertheless, multiple objects in an image can be handled and the solution is much cheaper than comparable DSP-/microcontroller/processor ones.

To handle images in 10 milliseconds is absolutely essential for real time embedded systems in IR seeker image processing. The detection of objects on an assembly line, for example, must be accomplished in reply times below 10 ms. It is implemented parallel processor architecture based on FPGA’s processing elements (PEs). This architecture is a SIMD approach, which means that exactly the same operations are carried out in parallel on each image pixel. This approach is a fast one and therefore very suitable for the processing of images. Important tasks required image processing, namely the determination of objects, their centre points and orientation. Closely related to this research work is the development of the parallel processing system which is for real-time image processing. It is composed of a high number of processors working in a SIMD fashion. In this research work parallel FPGA architecture the moments are used to determine characteristics of the detected objects. The mapping in parallel FPGA architecture.
4.1.1 Moments

Moments are a concept derived from statistics. They represent concrete physical characteristics and can be used to determine specific features in binary images like the center points or the orientation of objects. Image processing tasks are focused mainly on the processing of binary images. It is assumed that the binary images as a prerequisite for the calculation of moments. Thus, the \((k,l)\)-th moment of a discrete binary image is defined as follows:

\[
\mu_{k,l} = \sum_{x} \sum_{y} x^k y^l b(x, y)
\]

It holds that \(b(x, y) = 1\), if a pixel is set at the coordinate \((x, y)\), otherwise \(b(x, y) = 0\).

Correspondingly the \((k,l)\)-th central moment is:

In this equation \((x_c, y_c)\) denotes the coordinate of the centroid of an object in the binary image. By a translation of the origin of the coordinate system into this centroid, the central moments become independent from the concrete position of the object in the binary image. If one assumes connected binary regions, the moments can be directly calculated out of the coordinates of the outline points. The optimal way to handle images is to assign one pixel to each PE so that the design is fully parallel. estimate that multiple objects can be detected by

means of moments under the following conditions:

1. The camera must provide line parallel pixel.
2. The surrounding rectangles for each object should not overlap.
This leads to an increased rate of processed images per second. The only drawback might be the number of required slices: about one million for design (3 slices per PE times 640 times 480 plus some additional logic). The central part of the design is the processor field containing \((N+2)(M+2)\) PEs (\(N\) and \(M\) are the width and the height of each partition). All parts of the design work in parallel. That means, while one part of the image buffer is receiving pixels from the camera, the other one is sending a completed partition to the PEs. Concurrently, the PEs receive and process the data of a partition. The output of the PEs is stored into a FIFO which is read for post-processing.

Approximately half a million clocks (341814 was the maximum in test scenario. So if assume a frequency of 100 MHz, the reply times would be under 5 ms allowing us to process 200 images in one second. Such or higher frequencies can only be reached within a Virtex FPGA. About 75 MHz can be typically reached with a Spartan-3 1000 FPGA. Hence, the performance here lies around 150 processed images per second, however, at a much lower price. Furthermore, the pipelined operating of image buffer together with the PEs enable a higher throughput. Parallel processor architecture is able to detect and process multiple separated objects simultaneously.

### 4.1.4 Sum of Absolute Difference (SAD) Algorithm

SAD (Sum of Absolute Difference) algorithm is heavily used in motion estimation which is computationally highly demanding process in motion picture encoding. To enhance the performance of motion estimation on a (Parallel Sum of Absolute Difference) P\_SAD Parallel processor, an efficient implementation of SAD algorithm on the P\_SAD processor is essential. SAD algorithm is programmed as a nested loop
with a conditional branch. In P_SAD processors, loop is usually optimized by software pipelining, but researches on optimal scheduling of software pipelining for nested loops, especially nested loops with conditional branches are rare. In this research, SAD is implemented with 64x64 byte implementation on vertex 4 family for high speed motion vector calculation in embedded environments. Motion estimation in embedded environments are such as High speed object tracking missile camera and digital camcorder etc. Embedded processors need more computing power as demand of processing ability is increasing. Currently, adoption of SAD architecture becomes the trend of high-performance DSP. SAD supports instruction level parallelism.

Most object tracking applications need motion picture encoding and decoding. Motion estimation is one of the most important processes in motion picture coding since achievement of highest motion estimation is accomplished. However, motion estimation is computationally highly demanding process, and SAD (Sum of Absolute Difference) algorithm is the most heavily computed part in motion estimation process. Thus, an efficient implementation of SAD algorithm is very important. SAD algorithm is usually implemented in a nested loop with conditional branch. The conditional branch in SAD algorithm is for exit from the nested loop when the computed SAD value exceeds the minimum SAD value among previously computed SAD values. P_SAD processors is implemented FPGA by utilizing hardware technique. However, researches on optimal scheduling of image data, especially nested loop with branch are used. Based on this optimal scheduling strategy, optimal implementation of SAD algorithm on FPGA is implemented. Through experiments, it is shown that the proposed P_SAD implementation performs better than other SAD
implementations, and that the code size of algorithm is small enough to be appropriate for embedded environments. In the past, sizable research works on the implementation of block matching have been done, but most of them have dealt with efficient implementation of the algorithm itself irrespective of processor architecture. Also, research works on optimal implementation and performance of motion vector have been reported, but most of them deal with overall S/W design and performance analysis on SIMD processors and P_SAD processors, but rather than optimization of specific algorithms on P_SAD architecture. Performance of the serial motion picture encoding algorithm is worse than the motion picture encoding using SAD algorithm with conditional branch to exit from the nested loop. For performance analysis of SAD implementations, it is used H.263 encoder where the SAD algorithm in motion estimation module will be replaced by various SAD implementations. This research FPFA used for a P_SAD. Through experiments on a standard video sequence, it is shown that SAD using the proposed P_SAD implementation is much faster than adopting any other implementations of SAD.

4.1.5 SAD ALGORITHM IMPLEMENTATION IN MAT LAB

READ IMAGES I1 and I2
SUM of ABS Values
ABS Differences
SUM of ABS DIFFERENCE
COUNTER BASED TRACKING

READ IMAGES I1 and I2

I1 = imread ('3.bmp'); I2 = imread ('5.bmp');

% ILC = imlincomb ( 0.8, I2, 0.8, I1, -130 );
SUM of ABS Values

\[ ILC = \text{imadd}\left(\text{abs}(I2/2),\text{abs}(I1/2)\right); \]

ABS Differences

\[ I4 = \text{imsubtract}\left(\ I1,\ I2\ \right);
I5 = \text{imsubtract}\left(\ I2,\ I1\ \right); \]

SUM of ABS DIFFERENCE

\[ ISUM = \text{imcomplement}\left(\text{imadd}\left(\ I4,I5\right)\right); \]

\begin{align*}
\text{subplot}\ (2,2,1) & \quad \text{imshow}\ (I1); \quad \text{title}\ ('Frame_1') \\
\text{subplot}\ (2,2,2) & \quad \text{imshow}\ (I2); \quad \text{title}\ ('Frame_2') \\
\text{subplot}\ (2,2,3) & \quad \text{imshow}\ (ILC); \quad \text{title}\ ('\text{imadd}\ (\text{abs}(I2/2),\text{abs}(I1/2))') \\
\text{subplot}\ (2,2,4) & \quad \text{imshow}\ (ISUM); \\
& \quad \text{title}\ ('\text{imcomplement}(\text{imadd}(\text{imsubtract}(I1,I2),\text{imsubtract}(I2,I1))))')
\end{align*}

\[ Figure\ 4.1.5.1\ A\ Contour-Based\ Moving\ Object\ Detection\ and\ Tracking \]
4.1.6 Intensity Contour Extraction

Method for extracting contours of moving objects consists of 4 steps:

1. Line restoration,
2. Line-based background subtraction,
3. Clustering, and
4. Active contours.

Edges are sometimes incorrectly masked because of the use of local regions for computing optical flow. These miscalculations make detected edges fragmented. For restoring these fragments, masked edges are restored as unmasked edges (i.e. moving edges), if they are connected to points of moving edges without including any cross-points. However, restored lines also include background edges which are incorrectly detected as the motion. The noise as described above can be eliminated by subtracting background edges of the previous frame (except for the reflection of non-rigid objects). The detected and restored line of the current frame is eliminated if the line belongs to a background line. Then nearest-neighbor clustering is used with respect to the distance and velocity. Contours of the clustered lines are extracted \(^{[1],[2]}\).

4.1.7 COUNTER BASED TRACKING

\[
I_3 = \text{rgb2gray}(I_1); \quad \% \text{ color to intensity conversion}
\]

\textit{Figure} (2)

subplot (2,2,1); imshow(I3 ); title ('imshow(I3 )');

subplot (2,2,3); imcontour(I3 ); title ('imcontour(I3 )');

\% MAX INTENSITY SEARCH FUNCTION

IMAX = max(max(I3)) - 20;
Frame 4.1.6.1 A complex scenario of SAD Algorithm

```matlab
subplot(2,2,2); imshow((I3-180)*5); title('imshow(I3-IMAX)');
subplot (2,2,4); imcontour(I3-IMAX); title('imcontour(I3-IMAX)');
```

Frame 4.1.7.1 A Contour-Based Moving Object segmentation

```matlab
subplot(2,2,2); imshow(B); imcontour(B); title('imcontour(B)');
subplot(2,2,4); imshow(I3-IMAX); imcontour(I3-IMAX); title('imcontour(I3-IMAX)');
```
Figure 4.1.7.2 Region of interest of an object

The algorithm try to maintain the track of a group as a single entity. The tracking algorithm attempts to maintain objects in the form in which they were instantiated, which is achieved by means of morphological operators. A merging operator deals with silhouette fragmentation and a partitioning function handles silhouette merging events. Sometimes the fragmentation of objects is so poor that a perfect track cannot be maintained. The algorithm is able to track poorly segmented objects and no prior models of size, shape or texture are used. This is consistent with the overall strategy of a self-organising system, were objects are tracked and their behavior is classified without a priori knowledge built into the system. The algorithm is extremely fast, as the elements used during the match process are simple macroscopic features of silhouette area, width, height and grayscale histogram.
4.1.8 Tracking Detected Objects

To solve the correspondence problem for detected objects in different frames, the similarity between an object of the previous frame and an object of the current frame is defined using estimated positions of lines by optical flow.

Figure 4.1.8.1 SAD Algorithm based Moving Object Detection, Filter and Projection Curves, And Centroid Calculation

Difference (Fr₁ - Fr₂) : After Frame Subtraction.
Particle Filter : After filtering and Thresholding.
IR Position : After Projection Curve Analysis.
Centroid : Location of the IR Object Projection.
Figure 4.1.8.2 Object tracking using Projection Curve Analysis

Figure 4.1.8.3 Point Object Tracking using Radon Transform for Theta = 0° and 90° X, Y positions after Left Shift and Right Shift of the Object.
Figure 4.1.8.4 Large Object movement analysis using Radon Transform (Shift)
Projections of moving object for 20 frames overlapping

Figure 4.1.8.5 Discrete Large Object Tracking using Radon Transform (Shift)
Projections of moving object for single frame. Change in Shape of curve and location of the Maxima clearly visible.
4.2 PARALLEL IMAGE PROCESSING ALGORITHMS

Parallel image processing systems can be classified into three categories:

a) computer-based dedicated systems,

b) computer-based general systems and

c) DSP-based systems.

According to this classification, analysis and comparison of many kinds of realization technologies and structure characteristics have been carried out. The key principle for the choice of architectures is the system scalability and algorithm complexity.

4.3 IMPLEMENTATION OF SIMD ARCHITECTURES

The Single Instruction Multiple Data (SIMD) concept is a method of improving performance in applications where highly repetitive operations need to be performed. SIMD is a technique of performing the same operation, be it arithmetic or otherwise, on multiple pieces of data simultaneously.

Traditionally, when an application is being programmed and a single operation needs to be performed across a large dataset, a loop is used to iterate through each element in the dataset and perform the required procedure. During each iteration, a single piece of data has a single operation performed on it. This is known as Single Instruction Single Data (SISD) programming. Loops are very inefficient, as they can iterate thousands of times. Ideally, to increase performance, the number of iterations of a loop needs to be reduced. One method of reducing iterations is known as loop unrolling.
This takes the single operation that was being performed in the loop, and carries it out multiple times in each iteration. The SIMD concept takes loop unrolling one step further by incorporating the multiple actions in each loop iteration, and performing them simultaneously. With SIMD, not only can the number of loop iterations be reduced, but also the multiple operations that are required can be reduced to a single, optimized action. SIMD does this through the use of vectors. A SIMD vector can be used as an argument for a specific instruction that will then be performed on all elements in the vector simultaneously.

If we are to use a SIMD processor to operate upon images having many more pixels than there are SIMD processing elements, then must serialize computation to some extent. One straightforward way is to break image operands up into smaller image sub frames or blocks. This breaks the operation into a sequence of array-sized, parallel computations. This leads us to divide image operands into overlapping sub frames. As the number of such sub frames increases, the time to communicate the original image grows, eventually overcoming the savings achieved by avoiding temporary results.

Hardware implementation of an algorithm developed to provide automatic motion detection and object tracking functionality embedded within intelligent seeker systems. The implementation is targeted at an Xilinx Vertex -5 FPGA making full use of the dedicated DSP and Image Processing resources. The Vertex 5 provides an embedded processor provides a platform for the tracking control loop and generic Pan, Tilt, Zoom, Focal Plane Array(FPA) interface. This research details with the explicit functional stages of the algorithm that lend themselves to an optimised parallel Image Processing hardware implementation. This implementation provides
maximum data throughput, providing real-time operation of the described algorithm, and enables a moving FPA to track a moving object in real time

**SIMD Architecture**

The Single Instruction Multiple Data (SIMD) concept is a method of improving performance in applications where highly repetitive operations need to be performed. SIMD is a technique of performing the same operation, be it arithmetic or otherwise, on multiple pieces of data simultaneously.

Traditionally, when an application is being programmed and a single operation needs to be performed across a large dataset, a loop is used to iterate through each element in the dataset and perform the required procedure. During each iteration, a single piece of data has a single operation performed on it. This is known as Single Instruction Single Data (SISD) programming. Loops are very inefficient, as they can iterate thousands of times. Ideally, to increase performance, the number of iterations of a loop needs to be reduced. One method of reducing iterations is known as loop unrolling. This takes the single operation that was being performed in the loop, and carries it out multiple times in each iteration.

The SIMD concept takes loop unrolling one step further by incorporating the multiple actions in each loop iteration, and performing them simultaneously. With SIMD, not only can the number of loop iterations be reduced, but also the multiple operations that are required can be reduced to a single, optimized action. SIMD does this through the use of vectors. A SIMD vector can be used as an argument for a specific instruction that will then be performed on all elements in the vector simultaneously.
Because of this, the number of values that can be loaded into the vector directly affects performance; the more values being processed at once, the faster a complete dataset can be completed. This size depends on two things:

1) The data size being used and
2) The SIMD implementation.

When values are stored in SIMD vectors and worked upon by a SIMD operation, they are actually moved to a special set of CPU registers where the parallel processing takes place. The size and number of these registers is determined by the SIMD implementation being used. SIMD makes use of multiple CPU functional units; independent functional units for arithmetic and Boolean operations that execute concurrently.

The SIMD implementation can be enhanced with pipelining the program instructions. Instruction pipelining is the decomposition of instruction execution into a linear series of autonomous stages, allowing each stage to simultaneously perform a portion of the execution process (such as decode, calculate effective address, fetch operand, execute, and store).

In order to accommodate in hardware the requirements for elaborating both high and low precisions data, appropriate data-paths have to be supported and extreme flexibility has to be guaranteed. Such hardware can be efficiently realized in Single Instruction Multiple Data (SIMD) fashion. When properly designed, SIMD architectures are able to efficiently elaborate the highest precision data and allow fine-grained parallelism to be exploited in processing data on lower precisions.
4.4 IR IMAGE PRE-PROCESSING ALGORITHM

This section provides image processing related to image enhancement. Images received through various infrared (IR) devices in many applications are distorted due to the atmospheric aberration mainly because of atmospheric variations and aerosol turbulence \(^{26}\) \(^{27}\). Hence, image enhancement is a very popular field in image processing. Enhancement aims at improving the visual quality of an image by reinforcing edges and smoothing flat areas.

These images have a special nature of large black areas and small details due to the absence of the appropriate amount of light required for imaging. So, the main objective is to reinforce the details to get as much details as possible.

A few algorithms suggested for the same are: \(^{28}\) \(^{29}\)

1) SAD
2) H.264
3) Radon Transforms

Motion estimation (ME) is a multistep process that involves not one, but a combination of techniques, such as motion starting point, motion search patterns, and adaptive control to curb the search, avoidance of search stationary regions, etc. The collective efficiency of these techniques is what makes a ME algorithm robust and efficient across the board.
4.5 SUM OF ABSOLUTE DIFFERENCE ALGORITHM

Sum of Absolute Differences (SAD) is a widely used, extremely simple video quality metric used for block-matching in motion estimation for video compression. It works by taking the absolute value of the difference between each pixel in the original block and the corresponding pixel in the block being used for comparison. These differences are summed to create a simple metric of block similarity of the difference image.

When compared to other metrics, SAD is an extremely fast metric due to its simplicity; it is effectively the simplest possible metric that takes into account every pixel in a block. Therefore it is very effective for a wide motion search of many different blocks. SAD is also easily parallelizable since it analyzes each pixel separately, making it easily implementable on FPGAs. Once candidate blocks are found, the final refinement of the motion estimation process is often done with other slower but more accurate metrics, which better take into account human perception. These include the sum of absolute transformed differences (SATD), the sum of squared differences (SSD), and rate-distortion optimization.

Motion estimation produces. At any block location \((x, y)\); the SAD criterion is defined as:

\[
\text{SAD}(x, y, i, j) = \sum_{l=0}^{M-1} \sum_{k=0}^{N-1} |A_{(x+l,y+k)} - B_{(x+i+l,y+j+k)}|
\]

... 4.5.1

where \(A_{(x+l,y+k)}\) and \(B_{(x+i+l,y+j+k)}\) indicate pixels of the current block and the reference frame, respectively.
The size of the block is $M \times N$ and SAD computation is performed in the search area location $(i, j)$ which is the displacement of the candidate block compared to the current block.

The candidate block yielding the minimum SAD value determines $MV = (i, j)$ for the location $(x, y)$.

SAD architecture can functionally be divided into three stages:

1. Absolute difference calculation, accumulation of absolute differences, and

### 4.5.1 Absolute Difference Calculation

The purpose of the first stage is to calculate absolute differences between the candidate and current MB pixels. Vassiliadis et al.\(^{30}\) introduce a unit customized for detecting and inverting the smaller one of the pixel values. The unit calculates inversion with one’s complement arithmetic. Hence, to obtain finally a proper two’s complement SAD value, the introduced inversion error has to be compensated afterwards by an additional correction term. Another absolute difference unit is presented by Jehng et al.\(^{31}\). The unit produces an absolute difference with a one’s complement adder surrounded by a few logic gates. The implementation is called Jehng’s absolute difference unit. The third conventional absolute difference unit is presented by Chen et al.\(^{32}\).

It is an improved embodiment of Jehng’s unit and it is referred to as Chen’s absolute difference unit. In the unit, the low performance end-around carry chain in one’s complement adder is removed and the possible one-bit error is compensated later by a correction bit.
4.5.2 Accumulation of Absolute Differences

In the second stage, the produced absolute differences are accumulated. Parallelization that is exploited in the pixel accumulation can vary from purely serial to fully parallel.

Jehng et al.\cite{jehng1993} present an accumulation unit to be used with Jehng's absolute difference units. The adder tree implementation is called Jehng's adder tree unit. Despite the additional correction bits, Chen's absolute difference units are also well suited for the adder tree. The correction bit accumulation is implemented by substituting the correction bits to the carry-in inputs of the adders in the tree. The modified tree is referred to as Chen's adder tree unit\cite{chen1993}. The accumulation of absolute differences can also be implemented with CSA tree unit\cite{chen1994,chen1995}.

The carry-save adder (CSA) tree compresses the incoming absolute differences to carry and sum vectors. The carry propagation is performed during the last stage with a fast adder. Designing the CSA tree structure with more complex calculation elements\cite{chen1996} than full- and half adders. Chen et al. in\cite{chen1997} present a recursive CSA tree for SAD computation. It is hereafter called Chen's compression array unit.

Besides absolute differences and correction bits, Chen's compression array is capable of compressing previously compressed carry and sum vectors which are fed back to the array. Contrary to the accumulation units presented above, Chen's compression array produces two output vectors which are added together in the minimum SAD determination stage.
4.5.3 The motion estimator

The motion estimator has two inputs: a macroblock (MB) from the current frame and a 48 X 48 pixel search area (SA) from the previous frame. The motion estimator finds the best matching block in the search area. The H.264/AVC standard does not specify how this should be implemented. A possible implementation is shown in the following pseudocode:

```plaintext
for (MB =0; MB < 1089; MB ++)
{
    for (pixel =0, SAD =0; pixel < 256; pixel ++)
    {
        SAD += abs ( MB [pixel] - SA [pixel]);
    }
    if (SAD < bestSAD)
    {
        bestSAD = SAD;
    }
}
```

Two nested loops are used to iterate over all possible comparisons. The comparisons are evaluated by using the sum of absolute differences (SAD). Encoding one second of a 176144 pixel video (99 macroblocks per frame) at 25 frames per second requires 689,990,400 SAD operations. The vector between the position of the macroblock in the current frame and the position of the best matching block in the previous frame is called the motion vector. The current macroblock and the best matching block are subtracted, shown in the top right of Figure 4.5.1.
Figure 4.5.1 SAD Implementation Frame Subtraction of 2 Frames

Figure 4.5.2 Projection Curve analysis for Finding Difference of IR Spot.
The result is a residual, which is transformed, scaled and quantized. The H.264/AVC standard allows for seven modes with variable block sizes: a 16 x 16 pixel macroblock can be divided in smaller blocks to yield a better compression efficiency. Noise is removed by applying a morphological operator to the difference image. The pixels that remain classified as foreground are collected into connected components and assigned unique identities. Examples of flight and segmented silhouettes are shown in Figure 4.5.1. Along with an identity, each object has an associated feature vector, the elements of which are area, width, height and a histogram of the grayscale distribution of object pixels. This feature vector is used to match objects from frame to frame, as described in detail in the following sections. The object tracker described here is a purely measurement based object-to-silhouette matching algorithm with morphological manipulation that deals with uncertainty in the segmentation algorithm. The philosophy of the tracking algorithm is motivated by general top-down assumptions of the types of unusual behavior and normal activity that the system will have to deal with.

Typically, conventional architectures execute the minimum SAD determination stage in two successive phases. In the first phase, two partial SAD values are added together to compute the current SAD value, whereas the resulting SAD value is compared to the minimum SAD value in the second phase. To speed up the operation, Chen et al. \cite{1321} present an enhanced minimum SAD determination unit here called Chen’s MV determination unit. It performs SAD value comparison without completing actual SAD values at all. SAD values are utilized for comparison purposes only and the MV is the output from the unit. Some novel algorithms for optimization of the SAD algorithm...
and its implementation have been proposed by Vanne, J. et al\textsuperscript{38} while some alternate FPGA implementation techniques have been suggested by Wong, S. \textit{et al}\textsuperscript{39}

4.6 IR IMAGE SEGMENTATION

Segmentation partitions the captured image into several disjoint object regions based on common uniform feature characteristics. One simple method is to segment the image based on colour by applying thresholds to each pixel. This is ideal for stream processing because thresholding is a point operation which can be implemented easily on the FPGA.

The goal of image segmentation depends on the type image and its application. Segmentation is a complex image-processing task. A generic segmentation algorithm might not produce good results for a given application, although the same is known to provide good results for some other applications.

In the absence of any preset criterion, image segmentation uses a simple homogeneity criterion and partitions the image into "puzzle pieces" so that content of each piece is uniformly the same in appearance. Several pieces together may form an object of interest. In this chapter, discuss first segmentation of colored images, and then present its application to infrared images. The first goal is to identify objects of interest in infrared images.

A simple goal is to identify different colors by analyzing the L*a*b* colorspace.
4.6.1 Detecting Edges

The detected edge gives a bright spot at the edge and dark areas everywhere else. The detected edge is the derivative of the edge. $T_t$ is the slope or rate of change of the gray levels in the edge. The slope of the edge is always positive or zero, and it reaches its maximum at the edge. Edge detection is often called image differentiation.

A good application of edge detectors is to enhance edges and improve the appearance of an original image. Detect the edges in an image and overlay these edges on top of...
the original image to accent its edges. Detecting edges is a basic operation in image processing. The edges of items in an image hold much of the information in the image. The edges tell where items are, their size, shape, and something about their texture. An edge is where the gray level of the image moves from an area of low values to high values or vice versa. The edge itself is at the center of this transition.

Any edge detector can be used to enhance the edges in an input image. Simply add the option of taking the edge detector result or a value from the input image. An interesting project would be to use the 9x9 Gaussian mask to detect the edges of large objects and then use these edges to enhance the input image. Edge takes an intensity image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere. edge supports six different edge-finding methods: The Sobel method finds edges using the Sobel approximation to the derivative. It returns edges at those points where the gradient of I is maximum. The Prewitt method finds edges using the Prewitt approximation to the derivative. It returns edges at those points where the gradient of I is maximum. The Roberts method finds edges using the Roberts approximation to the derivative. It returns edges at those points where the gradient of I is maximum.

Figure 4.6.2 Edge detector is to enhance edges and improve the appearance
The number of masks used for edge detection is almost limitless. Researchers have used different techniques to derive masks and then experimented with them to discover more masks. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges. The parameters you can supply differ depending on the method you specify. If you do not specify a method, edge uses the Sobel method.

If a block meets the criterion, it is not divided any further. If it does not meet the criterion, it is subdivided again into four blocks, and the test criterion is applied to those blocks. This process is repeated iteratively until each block meets the criterion. The result can have blocks of several different sizes. The first edge detector is the homogeneity operator which uses subtraction to find an edge. Subtraction in a

Figure 4.6.3 Quadrant decomposition divides a square image into four equal-sized square blocks, and then tests each block to see if it meets some criterion of homogeneity.
homogeneous region (one that is a solid gray level) produces zero and indicates
an absence of edges. A region containing sharp edges, such as area has a large
maximum. Thresholding at 30 to 50 for a 256 gray level image gives good results.
The next edge detector is the difference operator, another simple operator that uses
subtraction. Recall that edge detection is often called image differentiation (detecting
the slope of the gray levels in the image). The difference operator performs
differentiation by calculating the differences between the pixels that surround the
center of a 3x3 area. Segmentation of the same frame. If segments instead of blocks
are used for motion estimation, motion boundaries can be obtained with pixel
accuracy.

4.6.2 Fuzzy Logic-Based Segmentation

Fuzzy thresholding, is used to extract objects of known dimensions, by separating
the foreground from the background. The method requires that the maximum and
minimum size of the object is known a priori. This assumption was made on the basis
that the size of the tank could be estimated from the knowledge of distance between
the tank and the camera onboard. The minimum size of the object is used for global
thresholding and to locate where in the image the object of interest. The maximum
size of the object is used for identifying ROI.

Once the ROI is identified, a fuzzy logic based refinement is performed in ROI. They
used this refined object to further identify the edges of the tank. They also assume that
there exists sufficient contrast between the foreground and background. It is assumed
that no knowledge about types, shape, distance or dimensions of object in the scene is
given. However it is assumed that there exist sufficient contrast between the
foreground and background. Since the size of the object is not known both global thresholding and ROI identification cannot be performed [42]. Thus there are two major problems, one is to determine global threshold and the other is to find the ROI around each object in the image. To avoid these limitations, it is first discussed some other global thresholding methods for infrared images, and then adapt one of them to expand [42]. The results are compared with the expanded method [42]. Fengliang Xu and Kikuo Fujimura [43] faced the above problem, where they considered identification of multiple pedestrian in night vision images. Otsu's method selects a threshold from the histogram following some discriminant analysis [44].

In this method, pixels are divided into two classes at the threshold point, which maximizes a discriminant measure. Discriminant function is the measure of separability of the resultant class. This method measures the separability of all the gray levels and finally chooses one that maximizes the separability as the proper threshold. Threshold based in Otsu's method can be used for thresholding bimodal histograms.

4.6.3 Adaptive Global Thresholding

One of the limitations discussed earlier is the problem of determining a proper threshold to process the histogram without any prior knowledge of the size of the main object in the image. Therefore, Sun-Gu Sun's approach is inapplicable for most surveillance images. To solve this problem, an alternative approach is proposed to compute threshold. A number of approaches have been purposed for histogram-based thresholding [45, 46, 47, 48, and 49].
The mode of interest always lies on the rightmost part of the histogram, because it pertains to brighter areas of the image, which should correspond to radiating objects in an IR image. This mode should always be separated from other modes regardless of its size.

If any gray level is selected on the brighter side as the threshold, it might not work suitably because the gray level assignment depends entirely on the relative temperature (radiation) difference between the foreground and background. Hence, it is necessary to determine a cluster (mode) in order to be able to separate foreground from background.

4.6.4 Adaptive Global Thresholding with Maxima-Based Detection

Sun-Gu Sun's approach [42] assumes that size of the object is known; it uses this information to separate background from the foreground. In absence of any such
knowledge, an adaptive method of thresholding is needed for images that might contain number of objects of interest. In the present method, the threshold is calculated by analyzing the histogram. This approach uses simplified bi-modal and unimodal thresholding with maxima based mode detection. The following assumption are made in the implementation:

Although object tracking is considered a mature field of research, there is a disturbing lack of uniformity in how results are presented by the community. Award-winning tracking papers rarely use the same data or metrics. This makes comparisons between different methods difficult and stifles progress. At the root of this problem is the lack of common data sets and performance measures.
**Figure 4.6.6** Multiple Objects detection using SAD Algorithm and Adaptive Global Thresholding Algorithm

**Figure 4.6.7** Multiple Objects Segmentation Intensity based Contour Algorithm
In order to define a framework for tracking evaluation, it is important to understand what qualities are essential to good tracking. To do so, it can be helpful to consider what constitutes a “golden” multi-object tracker. One could argue that a good tracker, in a real-life situation to identify objects well - track individual objects consistently over a long period of time, track objects in spite of distraction (occlusion, illumination changes, etc.), accurately estimate task-specific object parameters (such as object velocity), be fast.

4.7 Radon Transform Projections

4.7.1 Target tracking using projection curves analysis

There are many tracking methods for different targets and for targets in different moving states. When the target is moving close to the camera, its image area in the field of view (FOV) will gradually increase. The target in this state is generally called an extended target. In a real-time imaging tracking system, it is hard to accurately locate an extended target because of the gray-level overlapping between the targets. The correlation tracking algorithm (CTA), based on image matching, is one of the most efficient technique for tracking an extended target.

Some of the most commonly used matching criteria are the mean absolute difference (MAD), the mean squared error (MSE), the normalized cross correlation (NCC), and the matching pel count (MPC). However, application to a real-time tracking system was limited because of the complex environment. Many researchers have provided object tracking algorithms to sum a verity of applications, including vehicle tracking, medical diagnosis, surveillance, and military applications under a cluttered background.
Background subtraction and background motion compensation are also commonly used for detecting and tracking objects. Its tracking capacity strictly depends on the accuracy of background modeling, and if often fails under noisy, complicated background or without any a priori information or any constraint with respect to the camera’s position or the object’s motion. Optical flow estimation could be an efficient approach for tracing objects but its application in real-time systems was limited due to high computational complexity and sensitivity to noise.

In order to degrade the computational complexity improve real-time tracking performance, a novel algorithm for detecting and tracking extended target using the projection curves analysis and correlation tracking based on maximum matching pel count (MPC) criterion is presented. First, the projection curves of the difference image of two consecutive frames are analyzed to fine the approximate areas of the moving target on the entire scenes. Then correlation tracing based on the improved MPC criterion is used for target tracing against a cluttered background. Experimental results show, compared to the conventional approaches, that the proposed algorithm is more robust, has higher precision, and has simplified computational complexity for tracking and extended target against a cluttered background.

4.7.2 Projection curve analysis

In Real-time it is used for detecting, tracking, and guiding aircrafts or panzers near the ground. Before the closed-loop tracking, it is need to capture the object in several previous frames of video taken by monitors as quickly as possible. The camera could be considered a stationary. Moreover, the high frame-rate camera used for this system is up to 70Hz, and the displacement of moving object with in consecutive frames is
less than a pixel. The difference image of two consecutive frames in video taken by static monitors in the initial stage for capturing the target shows the areas of moving targets because the background is almost invariable between consecutive frames. The method using frame difference image has simplified the computational complexity.

In this research it is supposed that $P_k$ $(k=1, 2, 3, \ldots)$ is the $k_{th}$ frame of the monitoring video with resolution of $M \times N$ pixel.

$$D_t = P_{t+1} - P_t$$

Is the difference image between time $t$ and $t+1$.

In addition, suppose that the target's gray level is always brighter than the background. If the pixel $P_t(i, j)$ belongs to a bright moving target in position $(i, j)$ at time $t$, and $P_{t+1}(i, j)$ belongs to the dark background in position $(i, j)$ at time $t+1$ then $D_t(i, j)$ should be positive. On the contrary, $D_t(i, j)$ should be negative. So in this case, define the difference image for positive and negative cases as follows.

$$D_{pt}(i, j) = D(i, j) \text{ if } D(i, j) > \tau \text{ and } 0 \text{ otherwise} \quad (1)$$

$$D_{mt}(i, j) = |D(i, j)| \text{ if } |D(i, j)| > -\tau \text{ and } 0 \text{ otherwise} \quad (2)$$

Where $\tau$ is a threshold. Suppose that $D_{pt}$ consists of those pixels belong to targets in $P_t$ and belonging to the background in $P_{t+1}$, thus in this case define

$$f_u(m) = \sum_{i=1}^{n} D_{pt}(i, j) \quad m=1,2,3,\ldots,M. \quad (3)$$

$$f_v(n) = \sum_{j=1}^{n} D_{pt}(i, j) \quad n=1,2,3,\ldots,N. \quad (4)$$

Where $f_u(m)$ is the horizontal projection curve and $f_v(n)$ is the vertical projection curve of $D_{pt}$. For $D_{pt}$ and $D_{mt}$, there is a total of four projection curves, which can be denoted with $f_{pu}(m), f_{pv}(n), f_{mu}(m)$ and $f_{mv}(n)$, respectively. In this research can analyze these curves as follows steps.
1. Image preprocessing; include noise reduction and curves smoothing.

2. \( f_{pu}(m) = \max_{m=1}^M f_{pu}(m) \); find \( m_1 \) and the range \((i_1, r_1)\) of the relative wave peak.

3. \( f_{nu}(m) = \max_{m=i_1}^r f_{nu}(m) \); find \( m_1 \) and the range \((i_1, r_1)\) of the correlation wave crest.

4. Remove the wave crest just found.

5. Repeat steps 2 and 3, until \( \max(\max(f_{pu}(m)), \max(f_{nu}(m))) < \varepsilon \)

6. do the same process for \( f_{pv}(n) \), and \( f_{nv}(n) \); then in this research can find all the possible areas of moving targets.

7. Repeat steps 2-6 in all possible areas until they could not be divided.

**Figure 4.7.1** The pairs of wave crest represent a possible moving target in projection curve analysis.

Figure 4.7.1 shows three object results of analyzing projection curves that illuminates a possible area of a moving target where \( y_1, y_6, x_1, x_6 \) denote the possible areas containing a moving target.

### 4.7.3 Correlation tracking based on MPC criterion

Each pixel in a search image is categorized as a matched pixel or a non-matched pixel according to the MPC criterion, which can be formulated as the following expression.
\[ T(i, j; x, y) = 1 \quad \text{if} \quad |I(i, j, k) - I(i+x, j+y, K+l)| \leq \text{th}, \]
And
\[ T(i, j; x, y) = 0 \quad \text{otherwise} \quad \text{(5)} \]

Where \( \text{th} \) is a predefined threshold, and \( I(i, j, k) \) is the intensity of the pixel at location \( (i, j) \) in a searching block in the \( k \)th frame. The motion estimation of the searching block \( B \) is given by

\[ \text{MPC}(x, y) = \sum_{(x,y) \in B} T(i, j; x, y) \quad \text{(6)} \]

And
\[ [\hat{x}, \hat{y}]^T = \underset{(x,y) \in B}{\max} \text{MPC}(x, y) \quad \text{(7)} \]

Where \( (\hat{x}, \hat{y}) \) is the estimation value of \((x, y)\), which gives the maximum number of the matched pixel \((\hat{x}, \hat{y})\) is.

### 4.7.4 Improved MPC Criterion

The traditional definition of MPC in Eq (5) efficiently can restrain the effect of noise. However, it counts the number of matching pairs of correlative pixel but does not consider the matching performance of their pairs, which also denotes the correlation of two image blocks. Therefore, change the definition of MPC in Eq (5) as follows.

\[ T(i, j; x, y) = \text{th}' - |I(i, j, k) - I(i+x, j+y, K+l)| \leq \text{th}' \]
if \[ |I(i, j, k) - I(i+x, j+y, K+l)| \leq \text{th}' \]
\[ T(i, j; x, y) = 0 \quad \text{otherwise} \quad \text{(8)} \]

Where \( \text{th}' \) is also a predefined threshold that is somewhat different from \( \text{th} \) in Eq (5), and its value is usually less that \( \text{th} \). This new definition of MPC considers the effect of matching performance of each matching pair and can improve the tracking precision.
4.7.5 Adaptive pixel threshold

Applying a constant threshold $th'$ to all cases in real-time environment is difficult. Therefore, an adaptive pixel threshold that can determine the most important portion of the computation in the change detecting for a frame is rather critical.

The threshold selection for image segmentation and motion detection was studied over 10 year ago. Two of the simplest methods for selecting adaptive threshold are based on combining the mean value with the variance of image data and the histogram-based approach.

However, they are usually used for single-frame images or stationary image processing. The motion estimation procedure requires a threshold for separating these two kinds of blocks. Shi and Xia presented a thresholding multi-resolution block matching algorithm using a predefined MAD value to filter out inefficient blocks before further block matching. Another method is a histogram-based approach to threshold selection is derived under the assumption that the histogram generated from the intensity difference between two gray-level frames contains three values combined with additive Gaussian noise. Both of these algorithms perform well in terms of the trade-off between time and distortion. However, the threshold values are predefined and obtained by offline computation. Many experiments must be conducted to obtain feasible constant threshold values for various video sequences.

In real-time tracking system, the pixel threshold $th'$ is just considered as a pixel counter for judging the correlation between two frames of video, rather than as the gray-level value of a pixel for segmenting image. Here $th'$ is a special pixel threshold tracking based on the MPC criteria.
Before searching a motion vector in the search area, a direct prediction uses the same block (M×N) in the reference frame. Here, the prediction error, termed the initial MAD (IMAD) for each block, is defined as

\[ IMAD = \frac{1}{M×N} \sum_{i,j,k} |I(i, j, k) - I(i, j, k-1)| \] ------ (9)

Where \( I(i, j, k) \) denotes pixel value at the \((i, j)\) position in the current frame \(k\) and \( I(i, j, k-1) \) represents the pixel value at the same position in the previous frame.

Then in this research can define the adaptive pixel threshold as \( th' = \lambda \times IMAD \),

Where \( \lambda \) is a coefficient that can be set like 0.8-1.2

### 4.7.6 Template updating

In practice, if in this research take the current frame at the optimal Matching position as the template for the next frame matching, the tracking point is easy to drift onto the background from the correct position because of the large motion or the abrupt change of the pixel intensity in some frame. With accumulating error during tracking, it is more likely to lose the tracking point of the target in FOV. Therefore, it is necessary to take account of updating the template by combining the previous template with current frame near the \((\hat{x}, \hat{y})\) position. This approach provides timely guidance for the tracking procedure, so that in this research obtain good tracking results.

The traditional method for template updating is generated by using an infinite impulse response (IIR) filter of the form.

\[ M_{k+1} = aI(k+1) + (1-a) M_k \]
Where $M_k$ is the previous template in the current position, $I_{k+1}$ is the block of the current search near the $(x, y)$ position, $\alpha$ is an adaptive weighted coefficient, which can be defined as $\alpha = \frac{d_{MPC}}{A(M)}$. $d_{MPC}$ is the MPC value between the previous template and the current searching block at the $(x, y)$ position, and $A(M)$ is the total pixel of the template i.e. The template area. This approach for updating the template may be invalid because of the abrupt change of the pixel intensity and partial occlusion for the target. In the present work propose a new approach is proposed for updating the template combining projection curves analysis, shown in Figure 4.7.2

![Figure 4.7.2 Block diagram of template updating during tracking](image)

The main step of target tracking for the $k_{th}$ frame in image sequences can be summarized as follows.
1. Search the best matching block (block 1) of the current template (template 1) in frame k.

2. According to the projection curves analysis in frame difference image of frame k-1 and k, in this research get a new possible area of the target that is correlative with template 1. Then template 2 can be determined by choosing suitable block in it.

3. Searching the best matching block (block 2) of template 2 in frame k.

4. Replace the current template (template 1) using the better one by comparing block 1 with block 2.

5. $K=k+1$; so back to step 1.

### 4.7.7 Experimental Results

In order to evaluate the performance of the proposed algorithm, a series of computer simulations has been conducted using image sequences from a real scene. The proposed algorithm is not sensitive for single noise and is especially efficient when overcoming the influence when it target was locally hidden, distorted, or of acutely varying illumination.

Moreover with very large-scale integration (VLSI) notice that the proposed method can be combined with the searching algorithms to solve the time-consuming problem in future work.
Figure 4.7.3 Projection curves analysis
4.7.8 Image Projections and the Radon Transform

The basic problem of tomography is given a set of 1-D projections and the angles at which these projections were taken, how do reconstruct the 2-D image from which these projections were taken? The first thing did was to look at the nature of the projections. RADON Compute Radon transform. The RADON function computes the Radon transform, which is the projection of the image intensity along a radial line oriented at a specific angle.

\[ R = \text{RADON}(I, \text{THETA}) \]

returns the Radon transform of the intensity image \( I \) for the angle \( \text{THETA} \) degrees. If \( \text{THETA} \) is a scalar, the result \( R \) is a column vector containing the Radon transform for \( \text{THETA} \) degrees. If \( \text{THETA} \) is a vector, then \( R \) is a matrix in which each column is the Radon transform for one of the angles in \( \text{THETA} \). If you omit \( \text{THETA} \), it defaults to 0 to 179. Matlab Function \([R, \text{Xp}] = \text{RADON}(...)\)
returns a vector \( \text{Xp} \) containing the radial coordinates corresponding to each row of \( R \). The radial coordinates returned in \( \text{Xp} \) are the values along the x-prime axis, which is oriented at \( \text{THETA} \) degrees counterclockwise from the x-axis. The origin of both axes is the center pixel of the image, which is defined as:

\[ \text{floor}((\text{size}(I)+1)/2) \]

4.7.9 MatLab Program for Radon Transform

TITLE
Radome IM Process
RGB to Gray conversion
AIMING LINE Marks
CIRCLE Centre
display the center
plot the circle at centre

FINDING MAXIMUM INTENSITY
calculate Max INtensity Position in X
calculate Max INtensity Position in Y

MAX INTENCITY POINT

TITLE

%Important Observation sum of (more no. of low intensity) pixes dominates than (less no of high Intensity) pixels i.e. No = 100, I = 25 is 2500 but No = 10, I = 100 is 1000 in array is less signficant in Radon Transforms Global thresh hold is must in IR Image processing

clear all
clf;

Radome IM Process

%RGBN =imread ('FIREING.bmp');
%RGBN =imread ('AAAALD.bmp');
RGBN =imread ('IR12.bmp');

RGB to Gray conversion

%Thresholding level 200 CONTRAST MULTPLY Factor 10
% RGB to Gray Conversion

%XN = rgb2gray(RGBN);
% Thresholding -200

% XN = rgb2gray(RGBN -200 );

% contrast Enhancement by factor 10
XN = 10 * (rgb2gray(RGBN -200 ));

IMSIZE = size (XN ); % Size of image MxN

% subplot (3,1,1)

Figure (1)

imshow(XN );

AIMING LINE Marks

hold on

X1 = [ 0.25*IMSIZE(2) 0.75*IMSIZE(2) ]; % X axis
Y1 = 0.5 * [ IMSIZE(1) IMSIZE(1) ]; % X axis
line (X1,Y1);

X2 = 0.5 * [ IMSIZE(2) IMSIZE(2) ]; % Y axis
Y2 = [ 0.25*IMSIZE(1) 0.75*IMSIZE(1) ]; % Y axis
CIRCLE Centre

\[ xc = 0.5 \times \text{IMSIZE}(1); \]
\[ yc = 0.5 \times \text{IMSIZE}(2); \]

display the center

\% plot(xc,yc,'yx','LineWidth',5);

plot the circle at centre

\% use parametric representation of the circle to obtain coordinates of points on the circle

\[ \theta = 0:0.01:2\pi; \]
\[ \text{radius} = 20 ; \% \sqrt{(xc^2+yc^2)} \]
\[ X\text{fit} = \text{radius}\times\cos(\theta) + xc; \]
\[ Y\text{fit} = \text{radius}\times\sin(\theta) + yc; \]

plot(Xfit, Yfit);

\[ \text{radius} = 50 ; \% \sqrt{(xc^2+yc^2)} \]
Xfit = radius*cos( theta) + xc;
Yfit = radius*sin(theta) + yc;
plot(Xfit , Yfit );

FINDING MAXIMUM INTENSITY

[RN0,xpN0] = radon(XN,0); % Radon Transform at 0 deg

calculate Max Intensity Position in X
[MaxRN0, IndRN0] = max(RN0) ;

[RN90,xpN90] = radon(XN, 90);% Radon Transform at 90 deg

calculate Max Intensity Position in Y
[MaxRN90 , IndRN90] = max(RN90) ;
disp (IndRN0 );
disp (IndRN90 );

( X, Y ) = ( 222, 240 )
MAXx INTENCITY POINT

% plot(IndRN0 ,IndRN90 ,'.yx','LineWidth',2);

Figure (2)
subplot (2,1,1)
hold on
plot (xpN0, RN0);
axis ( [-0.5*IMSIZE(1) 0.5*IMSIZE(1) 0 MaxRN0 ] );
title ('Y Coordinate ')
hold off
subplot (2,1,2)
hold on
plot (xpN90, RN90);
axis ( [-0.5*IMSIZE(2) 0.5*IMSIZE(2) 0 MaxRN90 ] );
title ('X Coordinate ')
hold off
END

Figure 4.7.5 FPGA Based Hardware for radon Transform
An FPGA based hardware system for Radon Transform in X and Y direction 300 frames/Sec of 1024 x 1024 pixels frames.

Figure 4.7.6 FPGA based Hardware for Radon Transform

4.8 H.264 BASED MOTION ESTIMATION

Motion estimation is a multistep process that encompasses techniques such as motion vector prediction, determination of search range and search patterns, and identification of termination criteria. Each of these techniques has several diversions that may suit a particular set of video characteristics. It would be hard to conceive a universal algorithm that can perform well for all kinds of video contents.

However, if important characteristics of a video sequence can be identified and utilized for adjusting various steps of motion estimation, one can design an adjustable algorithm that can tune its parameters to suit the video at hand. This stage predicts the...
motion vector field (MVF) from the previous coded frame, and clusters macroblocks into background and foreground regions based on the predicted MVF.

Automatic detection and tracking of moving objects are the fundamental tasks of many video-based surveillance systems. Higher level security assessment and decision making procedures rely upon these essential video analysis tasks. Robust motion detection and object tracking provide the basis for detection of increased activity, entry into a restricted area, detection of objects left behind, tracking of optical flow against established motion patterns, and other similar surveillance requirements.

Using Intel™, H.264 software running on an Intel Pentium™ III 1.0 GHz general-purpose CPU with 512 MB of memory, achieving H.264/AVC SD with a main profile encoding solution would require approximately 1,600 BOPS (billions of operations per second). [Xilinx, Inc.] Courtesy Xilinx inc by Wilson C. Chung, Senior Staff Video and Image Processing Engineer Xilinx, Inc., wilson.chung@xilinx.com

The recently standardized H.264/MPEG-4 AVC video coder [50]. This upcoming standard is called to play an important role in the broadcasting market since it provides advances in digital video implementations in terms of bit rate reduction, transmission resiliency and video quality.

However, the impressive coding efficiency of H.264/AVC comes at the expense of significantly increased algorithmic complexity compared to existing standards, which has limited the availability of cost-effective, high-performance solutions Vaughn [51].
In fact, most of the existing real-time encoders for H.264/AVC are implemented on a DSP platform due its software flexibility for being upgraded, relatively low software development cost, and time-to-market reduction. Motion estimation in video coding standards, such as H.264/AVC, is considered to be the most time consuming encoding module. Motion estimation is generally performed on a 16x16 block, although in H.264/AVC, 7 different block sizes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4) are allowed resulting in 16 different functions to be implemented using a standard assembly (SA) description.

The Sum of Absolute Differences (SAD) algorithm consumes most of the encoding time either in the inter-coding for motion estimation or intra-coding module. To optimize the implementation of the motion estimation algorithm, the SAD engine needs to be optimized. SAD engine can be implemented on the C64 DSP from TI thus making use of C64 set of instructions to optimize the execution of the algorithm. TMS320C64 DSP is the most suited and architecture for multimedia applications.

A 1,000Hz visual feedback using the CMOS+FPGA vision. It is required to obtain positional and angular signals around 1,000Hz to control a mechanical system. A vision sensor must obtain visual features of a target object, synchronizing its sampling rate to the sampling rate of the control. Thus, need 1) image capturing over 1,000Hz with high resolution, 2) visual feature computation at the capturing rate, and 3) visual feature transmission to a control system with little delay.
Figure 4.8.1  Concept of SAD in H.262 Encoder working principle

Figure 4.8.2  PCI System H.262 Encoder H/w development

Figure 4.8.3  An FPGA-Vertex-II USB based embedded Object Tracking System
4.9 SYSTEM - ON - CHIP IMPLEMENTATION

System-on-a-chip or system on chip (SoC or SOC) refers to integrating all components of a computer or other electronic system into a single integrated circuit (chip). It may contain digital, analog, mixed-signal, and often radio-frequency functions all on one chip. A typical application is in the area of embedded systems. FPGA SOC is a SOC implemented in FPGA. It is fully flexible, very useful for specific applications like video processing and is the very first step towards developing Application Specific ICs (ASIC). Besides application-specific digital logic, the FPGA SOC contains Processors (soft-core, diffused), DSP blocks, Block-RAM, PLLs etc. Since FPGA can make use of parallelism for implementing image processing algorithms and has been shown to very suited for image processing, it is natural that a System on Chip solution would be based on FPGA SoC.

Figure 4.9.1 FPA SOC

The H.264 Motion Estimation is a fully functional netlist implemented on a Xilinx® FPGA. The Motion Estimation core accepts input parameters and macroblocks and generates output motion vectors and Sum of Absolute (SAD) values, in accordance with the Moving Picture Experts Group (MPEG).

Motion estimation (ME) requires the associated portion of the reconstructed frame to be available before any processing can start. The Reference frame is stored in the
external memory and is accessible through an external memory controller. One port is reserved for writing the reconstructed frame, and the user is responsible for loading it prior to starting the motion estimation process. The second port is reserved for the motion estimation reading the necessary data from the external memory. Figure 1 shows the H.264 Motion Estimation architecture.

The Motion Estimation core requires the user to provide a list of initial motion vectors where the searches will be performed. The Motion Estimation module is responsible for performing the searches around the provided location, as well as for the search area data management. The motion estimation algorithm uses the input information such as macroblock location for processing the current macroblock.

<table>
<thead>
<tr>
<th>FPGA Family</th>
<th>Clock FMax</th>
<th>Macroblock Throughput (MB/Sec)</th>
<th>External Memory Bandwidth (Gbps)</th>
<th>Max Limits Supports up to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spartan-3A DSP speed grade -4</td>
<td>130 MHz</td>
<td>108,000</td>
<td>1.65</td>
<td>720P@30</td>
</tr>
<tr>
<td>Virtex-4 speed grade -12</td>
<td>225 MHz</td>
<td>200,000</td>
<td>3.1</td>
<td>720P@50</td>
</tr>
<tr>
<td>Virtex-5 speed grade -3</td>
<td>275 MHz</td>
<td>244,800</td>
<td>3.74</td>
<td>1080P@30, 1080i@60</td>
</tr>
</tbody>
</table>

4.9.1 H.264 Motion Estimation System Overview

The Reconfigurable Architecture is domain-specific, not general-purpose. Therefore need to identify specific applications that will be targeted by the Reconfigurable
Archtectrue. In this research work, we choose H.264 for the target application, since it requires a lot of parallelized computation. The H.264 Motion Estimation core is intended to be used in a video encoding setting where motion estimation is key to system performance. The motion estimation core requires connection to an external frame buffer to perform block matching and provide the SAD value to be used in rate distortion criteria for best mode selection between inter and intra modes in an H.264 encoder. The system diagram of Figure 4.9.2 gives a high-level view of how the core should be connected into the encoder system. The operation of the core proceeds with first the reference frame being written to external memory. Upon completion of writing the reference frame, the first macroblock can be written to the core. This triggers a frame update where frame parameters are read into the core for processing of the frame. The core processes k-by macroblock with a latency of six macroblocks.

However, among the several new features which are introduced by H.264/AVC the Motion Estimation (ME) process is highly computationally intensive than traditional algorithms and count for about 80% of the total computation complexity. ME process exploits temporal correlation between adjacent frames in a video sequence to reduce the data inter-frame redundancy. In the ME, the current frame of video sequence is divided to Macroblocks (MB) and for each of them a best matched MB in the previous processed frames is searched within the search area. This best matched MB is selected as the MB with lowest coding cost. Then, the differential between the current MB and the best matched MB is coded to achieve high coding efficiency. Different from the traditional motion estimation algorithms H.264/AVC support seven kinds different block size motion estimation which are from 4x4 pixels to 16x16 pixels block. It also extends the concept of integer pixel motion estimation to
half and quarter pixel motion estimation. Furthermore, multiple reference frames are available to search more concise motion vectors. Another Rate Distortion Optimization (RDO) algorithm is adopted by H.264/AVC. It is an exhausted pre-coding loop for each mode of INTRA and INTER type sub-blocks to select the most efficient coding mode. These new techniques highly improved the coding efficiency as well as multiple times coding complexity.

4.9.2 H.264 Motion Estimation Engine Description

The H.264 Motion Estimation core computes the sum of absolute difference (SAD) for a set of 120 search locations within a 112 x 128 search window for 8 x 4 blocks. The search locations are determined by a set of 10 seeds that are provided by the user and the 4 x 3 region to the right and down from each seed. The core provides as output the 120 SAD calculated values and the motion vectors. In addition, the coded block pattern is computed for a macroblock and the best motion vector for each sub-block is provided. The functional inputs and outputs to the H.264 Motion Estimation Core are:

- Inputs:
  - New Macroblock
  - Parameters for the New Macroblock
  - Macroblock location \((x_i, y_i)\)
  - Motion Vector predictors
  - Reference Frame pointer
  - Reference Frame

- Outputs
  - Output Parameters
Figure 4.9.2 is a diagram of the H.264 Motion Estimation architecture. Motion estimation requires the associated portion of the reconstructed frame to be available before any processing can start. The Reference frame is stored in the external memory and is accessible through an external memory controller. One port is reserved for writing the reconstructed frame, and the user is responsible for loading it prior to starting the motion estimation process. The second port is reserved for motion estimation reading the necessary data from the external memory. The motion estimation is delivered as a netlist.

The Motion Estimation core requires the user to provide a list of initial motion vectors where the searches will be performed. The Motion Estimation module is responsible for performing the searches around the provided location, as well as for the search area data management. The motion estimation algorithm uses the input information such as macroblock location for processing the current macroblock.

The motion estimation core uses a sliding window module to load the information required by the current macroblock. The core allows a total search window of 128 x 128. However, to enable parallelism between search module and fetching data from the memory controller, one column is always used to load data. This makes the search window available for search 112 x 128 ((128-16) x 128). The external memory bandwidth required can be calculated based on following information: the sliding
window controller requires refreshing of one column of macroblocks in the search area. The bandwidth formula is: Where is the required memory bandwidth, is the total number of macroblocks from a frame and if the video frame rate in frames per second.

The factor 8 is attributed to the fact that the sliding window height is 8 macroblocks high and the algorithm requires one column of MB refresh for each motion estimated macroblock. For example, a 720p video sequence processing requires the following bandwidth: 720p => 3600 macroblocks, frame rate 30 fps, and it results in 210.9375 Mega Pixels per second => 1.7 Gbps as throughput requirements for the external memory.

![Figure 4.9.2 Motion Estimation Core on FPGA](image)

*Figure 4.9.2 Motion Estimation Core on FPGA*
REFERENCES


[17] Fatemi, Corporaal, Basten, Jonker, Kleihorst, "Implementing face recognition using a parallel image processing environment based on algorithmic frames"


211


[38] A High-Performance Sum of Absolute Difference Implementation for Motion Estimation, Jarno Vanne, Eero Aho, Timo D. Hämäläinen, and Kimmo Kuusilinna


[55] (1) Amir H. Farrahi,(2) Majid Sarrafzadeh, "TDD: A Technology Dependent Decomposition Algorithm for LUT-Based FPGAs", (1) IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, (2) ECE Department, Northwestern University, Evanston, IL 60208.

[57] Pawan Deshpande, "Decoding Algorithms for Complex Natural Language Tasks", Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Masters of Engineering in Computer Science and Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY, January 2007.


[62] V. Brost, F. Yang, M. Paindavoine, "THE RAPID PROTOTYPING EXPERIENCES OF IMAGE PROCESSING ALGORITHMS ON FPGA", 216

[64] Anthony Edward Nelson, “IMPLEMENTATION OF IMAGE PROCESSING ALGORITHMS ON FPGA HARDWARE”, Thesis Submitted to the Faculty of the Graduate School of Vanderbilt University in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Electrical Engineering, May 2000, Nashville, TN.


Eric Cigan, Narinder Lall, “Integrating MATLAB Algorithms into FPGA Designs”, (1) Product Marketing Manager AccelChip Inc. eric.cigan@accelchip.com, (2) Senior DSP Marketing Manager, Xilinx, Inc.


[69] Jonathan Owensa, Andrew Hunterb & Eric Fletchera, “A Fast Model-Free Morphology-Based Object Tracking Algorithm”, aSchool of Computing & Technology, University of Sunderland, UK, bDepartment of Computer Science Durham University, UK.


[71] Chaur-Chin Chen, “On the Selection of Image Compression Algorithms”, Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan.


[74] Lazaros Nalpantidis1, Georgios Ch. Sirakoulis2 and Antonios Gasteratos1, “Review of stereo matching algorithms for 3D vision”, Democritus University of Thrace, Department of Production and Management Engineering, GR-67 100
Xanthi, Greece 2Democritus University of Thrace, Department of Electrical and Computer Engineering, GR-67 100 Xanthi, Greece.