Object Tracking on FPGA-based Smart Cameras using Local Oriented Energy and Phase Features

Ehsan Norouznezhad, Abbas Bigdeli, Adam Postula, Brian C. Lovell
NICTA, PO Box 6020, St Lucia, QLD 4067, Australia, and
The University of Queensland, School of ITEE, QLD 4072, Australia
ehsan.norouznezhad@nicta.com, abbas.bigdeli@nicta.com
adam@itee.uq.edu.au, lovell@itee.uq.edu.au

ABSTRACT

This paper presents the use of local oriented energy and phase features for real-time object tracking on smart cameras. In our proposed system, local energy features are used as spatial feature set for representing the target region while the local phase information are used for estimating the motion pattern of the target region. The motion pattern information of the target region is used for displacement of search area. Local energy and phase features are extracted by filtering the incoming images with a bank of complex Gabor filters. The effectiveness of the chosen feature set is tested using a mean-shift tracker. Our experiments show that the proposed system can significantly enhance the performance of the tracker in presence of photometric variations and geometric transformation. The real-time implementation of the system is also described in this paper. To achieve the desired performance, a hardware/software co-design approach is pursued. Apart from mean-shift vector calculation, the other blocks are implemented on hardware resources. The system was synthesized onto a Xilinx Virtex-5 XC5VSX50T using Xilinx ML506 development board and the implementation results are presented.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

Object Tracking, Smart Camera, Gabor Filters, Mean-Shift Tracking, Local Features, FPGA

1. INTRODUCTION

Object tracking is one of the most important tasks in multi-camera surveillance networks. It can be a critical component in wide range of higher level surveillance tasks. Two approaches can be taken to track a target across multiple cameras. The first approach will be to centrally track target of interest across multiple cameras. In this approach, all the cameras will send videos to the central processing unit, having no track of targets in their field of view. This approach requires very high computational cost because the central processing unit should keep track of targets of interest in all the different fields of views monitored by the cameras.

Another approach will be to use smart cameras with object tracking modules embedded on the camera. In this approach, each smart camera track targets of interest in their field of view and only sends these coordinates to the central processing unit. Hence, knowing the geometry of the smart cameras, the central processing unit would be able to keep track of targets of interest across multiple cameras with much less computational power. Although object tracking is usually associated by high computational cost, thanks to recent advances in silicon technology, the computing power available in embedded devices is sufficient to perform computer vision tasks which are highly computationally and data intensive.

In this paper, we present a robust object tracking module for smart cameras. Object tracking is composed of two main components, the object representation and object tracker. For object representation, there are a number of spatial and temporal feature sets which can be used to represent an object. Features like color, shape, texture, and motion can be utilized as a feature set for object tracking. The chosen feature set plays an important role in the overall performance of the object tracking system. Therefore there is a tendency toward using more discriminative feature sets. In general, feature sets which can better discriminate objects from the background and from one another, can improve the overall performance of object tracking.

Color is the most popular feature set which is used for object tracking. Perhaps the main reason behind using color as a feature set for object tracking is that it does not require extra processing to extract while other feature sets such as motion or texture require extra computing to extract. However, the problem with color is that it is sometimes the case that color is not a discriminative feature, e.g. a foreground object which has the same color with a background or another foreground object. In these cases, even if a powerful tracker is used, the tracker will fail because the chosen feature set was not discriminative.

However, using feature sets other than color are usually associated with extra computational cost. Therefore in applications where real-time performance is required, it is usually not feasible to use other feature sets such as motion or

ICDSC 2010 August 31 – September 4, 2010, Atlanta, GA, USA

texture. It becomes even more challenging in the case of smart cameras and embedded vision systems with limited computational and memory resources.

In this paper, we propose using local oriented energy feature sets for real-time object tracking on smart cameras. Our experiments show that using this feature set can increase the robustness of the system particularly when color feature set fails. The local oriented energy features are extracted by filtering the incoming images with Gabor filters across multiple channels. This requires 2D convolution of incoming images with multiple Gabor filters. Due to the high computational cost, real-time performance could not be achieved with software-based implementations alone. Therefore in this paper, we propose a hardware/software architecture which can achieve real-time performance on certain image sizes and frame rates. Furthermore to improve the robustness of the system, we estimate the motion pattern of the target region. This is done by using local phase outputs of Gabor filters. We use phase-based optical flow technique to estimate velocity components and motion vectors of the target region.

The rest of the paper is organized as follows: in Section 2, we review some of the works in embedded object tracking. Also we review the works which used Gabor features for object tracking. An overview of Gabor filters is given in Section 3. Section 4 describes the proposed algorithm. Experiments are explained in Section 5. Section 6 describes the proposed architecture. The FPGA implementation results are given in Section 7. This is followed by Conclusion in Section 8 and discussion and future works in Section 9.

2. RELATED WORK

This section reviews the works in two different categories. In the first part, we review some recent works which have been done in object tracking on embedded devices. Our main purpose is to provide a brief survey of the proposed algorithms and architectures for real-time embedded object tracking systems. In the second part, we review the works which used Gabor features for the purpose of object tracking.

Jason Schlessman et al. [20] implemented an object tracking on a hardware/software co-design basis. In their system, object tracking is composed of three sub-systems; background subtraction using Mixture of Gaussian (MoG) followed by Optical flow based on KLT algorithm and object tracking. As optical flow was the most computationally intensive part of their system, encompassing 56 percent of execution time, it was implemented on hardware while background subtraction and object tracking were implemented in software.

Felix Muhlbauer et al. [19] presented a dynamic reconfigurable system for object tracking. They pursued a hardware/software co-design approach as well. The proposed system has basically three main stages; feature detection, feature selection and feature tracking. The first two stages are implemented on reconfigurable hardware while the feature tracking is implemented on Microblaze soft core processor.

Fabio Dias et al. [6] implemented an embedded template tracking system on their FPGA-based Smart Camera. They used template matching method as a tracking technique. Sum of absolute Difference (SAD) algorithm is chosen for template matching.

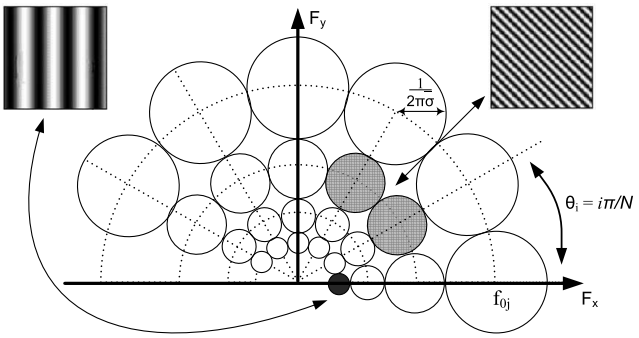


Figure 1: Gabor Filters at different channels to extract various texture Features (N is number of Filters)

Matteo et al. [21] proposed and evaluated the performance of two different FPGA-based embedded object tracking system. In the first system, color and blob tracking technique is used as feature vector and tracker respectively. The second system extracts motion information using highly parallelized optical system.

Y. Yamaoka et al. [22] proposed a real-time object tracking system based on image segmentation and pattern matching. It is basically composed of two stages. In the first stage, objects are extracted from images. Using a simple feature space, objects are tracked using a pattern matching technique.

Jung UK Cho et al. [2] proposed a real-time object tracking system using a Particle filter. In this work, the Particle filter is modified to be efficiently implemented on reconfigurable hardware.

On the other hand, there are very few works which used Gabor features for object tracking. Perhaps the main reason is that extracting Gabor features is usually associated with high computational cost. Chao He et al. [12] proposed an object tracking method for object-based video processing systems using Gabor Wavelet Transform (GWT) and 2-D golden section algorithm. Kevin Cannons et al. [1] proposed using spatio-temporal oriented energy features for object tracking. In their proposed method, the incoming images are filtered with spatio-temporal oriented filters, which are extremely computationally intensive.

All the aforementioned works in real-time object tracking on embedded platforms exploit feature sets other than texture, Gabor and local oriented energy features. On the other hand, the works which use Gabor features or local oriented energy features, do not propose a system which could achieve real-time performance. To the best of our knowledge, no work exists which uses Gabor feature space for embedded object tracking. Also we are not aware of any works which integrates local energy and phase as a solution for a computer vision application. Hence our proposed system is novel at two levels; it uses both spatial oriented energy features as well as the local phase features for object tracking on smart cameras and embedded vision systems in real-time.

3. GABOR FILTERS

There are different types of Orientational filters which can be used to extract local oriented energy features [16]. Gabor filters and second derivative of Gaussian are the most

popular ones [7]. In our system, the Gabor filter is used to extract local oriented energy feature sets mainly due to the fact that Gabor filter's parameters can be easily tuned to extract information lying in a specific frequency and orientation. Also in regards to hardware implementation, it can be efficiently implemented on reconfigurable hardware since it is a 2D convolution based operation,

The Gabor filter was originally introduced by Dennis Gabor in 1946 [10]. It has been widely used in facial recognition [17], iris recognition [18], fingerprint recognition [14] and texture segmentation [13] applications. The two-dimensional Gabor filter is defined as a multiplication of a complex sinusoid with a Gaussian envelope (Eq. 1):

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp -[\frac{(x^2+y^2)}{2\sigma^2}] \exp [j2\pi f_0(x' \cos\theta + y' \sin\theta)] \quad (1)$$

where

$$x' = x\cos\theta + y\sin\theta \quad (2)$$

$$y' = -x\sin\theta + y\cos\theta \quad (3)$$

In this equation, f_0 represents the central frequency of Gabor filter, θ is the orientation of the filter and σ is the standard deviation of the Gaussian window. These parameters can be finely tuned to extract any texture information. This is carried out by filtering across multiple channels with different frequencies and orientations. For illustrative purposes, a sample textures for different channels of Gabor filter is shown in Fig 1.

4. PROPOSED ALGORITHM

The process of object tracking using local oriented energy and phase features involves the following steps:

- Decomposition of incoming images using bank of complex Gabor filters;
- Extract local oriented energy features for every single pixel in the image;
- Generate a feature histogram for the target and candidate region;
- Find the best candidate for the target by computing the mean-shift vector;
- Estimate the motion pattern of target region using local phase information;
- Displace the search area using target's coordinates and its motion pattern information.

4.1 Multi-channel Gabor Filtering

The first stage of the algorithm is to convolve the incoming images with a bank of complex Gabor filters. The real and imaginary components of Gabor filter are tuned into multiple channels with different frequencies, scales and orientations. The tuning is done in a way to uniformly cover the spatial-frequency space. The selection of parameters is basically inspired from texture segmentation algorithms in which the parameters are tuned so that texture information can be extracted as much as possible [13]. In current implementation of the algorithm, we set the number of orientations to four, separated by 45 degrees. The frequencies are

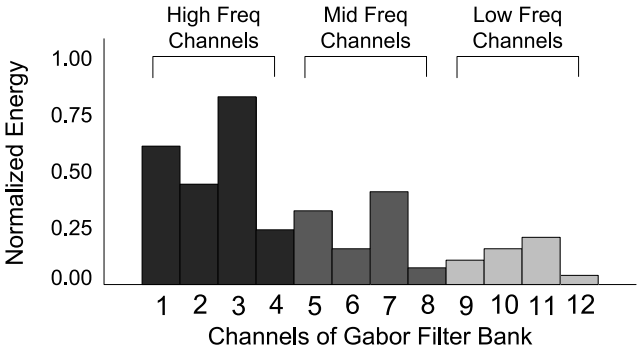


Figure 2: Sample Feature Histogram of target region in Fig. 3(i)

$4\sqrt{2}$, $8\sqrt{2}$ and $16\sqrt{2}$. It should be noted that the frequency bandwidth, which is $log_2(f_2/f_1)$ is set to 1 octave, where f_2 and f_1 are the aforementioned frequencies. Also the lower frequencies, e.g. $1\sqrt{2}$ and $2\sqrt{2}$ were discarded since these frequencies capture spatial variations that are too large to correspond to texture. This means, there are 12 channels in total with 4 different orientations and 3 frequencies and scales. This requires 24 times 2D convolutions for real and imaginary components of Gabor filter.

4.2 Local Oriented Energy Features

The second stage of the algorithm is to compute the local oriented energy features. The output of multi-channel filtering stage will be a multi-dimensional complex Gabor feature vector for every single pixel of the image. The local oriented energy features are computed using the real and imaginary components of complex Gabor feature. Hence, the local oriented energy features for every single pixel in the image can be computed as follows:

$$W_{Re,j}(x,y) = I(x,y) * G_{Re,j}(x,y) \qquad j:1,2,...,M \quad (4)$$

$$W_{Im,j}(x,y) = I(x,y) * G_{Im,j}(x,y) \qquad j:1,2,..,M \quad (5)$$

$$E_j(x,y) = Sqrt(W^2_{Re,j}(x,y) + W^2_{Rm,j}(x,y)) \qquad j:1,2,..,M \quad (6)$$

Where, $I(x,y)$ is the intensity value of pixel at coordinates (x,y), $G_{Re,j}(x,y)$ and $G_{Im,j}(x,y)$ are the real and imaginary components of complex Gabor filter across channel j respectively, M is number of channels and $*$ denotes Convolution. $E_j(x,y)$ is the local oriented energy value of pixel (x,y) across channel j. It should be noted that for every single pixel, M local oriented energies are extracted.

4.3 Feature Histogram

The third stage of the proposed algorithm involves computing the feature histogram for the target and the search area. Once local oriented energy features are extracted for all the pixels inside the target region and search area across different channels, the feature histogram can be computed. The histogram is calculated using local oriented energy values (E_j) in every single channel as shown in Eq. 7. The number of bins in histogram is equivalent to number of channels. Filtering at M different channels will lead to M bins for the histogram. In our current implementation, we use 12

channels, so that the number of bins is 12. For illustrative purposes, a sample feature histogram of a target region in Fig. 3 (ii) is shown in Fig. 2. There are 12 bins, divided into 3 main parts, representing the high, mid and low frequencies. As it is shown in this histogram, the energy values in high frequency bins have higher values for this particular target.

$$E_j = \sum_{i=1}^{N} E_{ij} \qquad j = 0, 1, ..., M \tag{7}$$

4.4 Mean-Shift Tracking

Mean-Shift tracker is chosen for this part of the system. It is mainly due to the fact that robustness and performance of mean-shift tracking algorithm is proven to be reasonably good [5, 4]. The mean-shift tracking algorithm is an iterative algorithm which uses the probability distribution of a target region and a candidate region to find the best candidate model in the next frame based on the target model in the current frame. A similarity function is used and maximized in each iteration to find the target candidate in the next frame. Mean-shift tracking algorithm is composed of four main stages which are briefly explained as follows.

4.4.1 Target Model

The target is represented by the rectangle region of width h_x and height h_y, centered at $\mathbf{0}$. The target model is computed as follows

$$\hat{q}_u = C \sum_{i=1}^{n} k\Big(\|\mathbf{x}_i^*\|^2\Big)\delta[b(\mathbf{x_i^*}) - u] \tag{8}$$

Where $\mathbf{x^*_{i\ i=1,2,...,n}}$ are the normalized pixel locations, n is number of pixels in reference image, $u = 1\ ...\ M$, M is number of bins, and $k(x)$ is the kernel function and δ is the kronecker delta function. The constant $\mathbf{C}$ is derived using the following equation:

$$C = \frac{1}{\sum_{i=1}^{n} k\Big(\|\mathbf{x}_i^*\|^2\Big)} \tag{9}$$

4.4.2 Target Candidate

The target candidate model is defined using Eq. 10:

$$\hat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\Big(\Big\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\Big\|^2\Big)\delta[b(\mathbf{x}_i) - u] \tag{10}$$

Where $\mathbf{x_{i\,i=1,2,...,n_h}}$ are the normalized pixel locations with n_h pixels centered at y in the current frame, C_h is the normalization constant with a following equation

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k\Big(\Big\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\Big\|\Big)^2} \tag{11}$$

4.4.3 Similarity Function

A similarity measure has to be used to define a distance between target and candidate models. Bhattacharyya Coefficient is used as similarity measure between the target and candidate model. The sample estimate between two discrete distributions of p and q can be computed using the following equation.

$$\hat{p}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} \tag{12}$$

4.4.4 Target Localization

To find the location corresponding to the target in the current frame, the distance should be minimized. Minimizing this distance is equivalent to maximizing Bhattacharyya coefficient. In this procedure, the kernel is recursively moved from the initial location of $\hat{\mathbf{y_0}}$ to the new location $\hat{\mathbf{y_1}}$ using Eq. 13. This process is done iteratively until the convergence or predefined maximum iteration number reached. In our current implementation, the maximum number of iteration is fixed to eight while the average number of iterations to find the best match is four.

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i \omega_i g\Big(\Big\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\Big\|^2\Big)}{\sum_{i=1}^{n_h} \omega_i g\Big(\Big\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\Big\|^2\Big)} \tag{13}$$

Where $g(x) = -k'(x)$ and ω_i is calculated by the following equation.

$$\omega_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q_u}}{\hat{p_u}(\hat{y_0})}}\delta[b(\mathbf{x_i}) - \mathbf{u}] \tag{14}$$

4.5 Target's Motion Pattern Estimation

To achieve more accurate results, the motion pattern of the target region is estimated to locate the search area in the next frame in proper position. The motion pattern of the target is estimated using phase-based optical flow technique. Phase-based techniques are shown to be one of the most accurate methods to estimate optical flow. This is mainly due to the fact that phase information is robust to changes in illumination, scale, orientation and speed [9]. The main idea behind using phase information to estimate motion is that the displacement in spatial domain is equivalent to phase shift in frequency domain. The phase information can be extracted by convolving the sequence of images with a bank of band-pass complex-valued quadrature-pair filters. Using phase information to estimate the motion vectors was introduced by D. J. Fleet et al. [8]. Their method is based on Spatio-temporally filtering of images, which is very computationally intensive and hard to implement on embedded systems. In our system, we use the method proposed by T. Gautama et al. [11]. Using their method, the image sequence is spatially filtered using a bank of quadrature pairs of Gabor filters. The output of the quadrature Gabor filter is complex-valued and we denote the phase component of the output by $\phi(x, y)$ (Eq. 15).

$$\phi_j(\mathbf{x}) = \phi_j(x, y) = Arctan(\frac{W_{Im,j}(x, y)}{W_{Re,j}(x, y))} \tag{15}$$

The velocity components are estimated using the temporal phase gradient (Eq. 16).

$$\phi_{t,j}(\mathbf{x}) = |\phi_j(\mathbf{x}, t + \Delta t) - \phi_j(\mathbf{x}, t)| \tag{16}$$

The directions of velocity components are orthogonal to the filter pair's orientation and the magnitude of the velocity components are estimated using the following equation:

$$V_{c,j} = \frac{-\phi_{t,j}(\mathbf{x})}{2\pi(f_x^2 + f_y^2)}(f_x, f_y) \tag{17}$$

Where V_c is the velocity component, ϕ_t is the temporal phase gradient, and f_x and f_y are horizontal and vertical frequencies of Gabor filter. For every filter pair j, the temporal phase gradient, $\phi_{t,j}(\mathbf{x})$ is computed from the temporal

sequence of its phase components. Also the phase values are unwrapped by adding or subtracting $(k.2\pi)$ if the temporal phase gradient exceeds π. Furthermore it should be noted that non-reliable phase values are rejected by estimating the phase nonlinearity as a confidence measure.

In our system, we assume that all the pixels inside the target region have uniform motion pattern. Therefore we use the average phase components of all the pixels in each channel and we compute the velocity component for the target region in that channel. The motion pattern of the entire target region is estimated using velocity components at different orientations. The full description of the motion detection block is beyond the scope of this paper.

4.6 The Search Area Displacement

The extent of search area around the target region is application dependent. In our current implementation, we chose to use location-adaptive fixed size search area. For the size of the search area, we use area equal to two times the target region. The location of the search area is updated using the target region coordinates and its motion patterns. The centre of the search window is updated using the following equations:

$$X_{cs} = X_{ct} + \alpha D_x \tag{18}$$

$$Y_{cs} = Y_{ct} + \beta D_y \tag{19}$$

where (X_{cs}, Y_{cs}) is the centre of search area, (X_{ct}, Y_{ct}) is the centre of target region, (D_x, D_y) is the displacement in horizontal and vertical directions defined by the motion vector, and α and β are the scaling factors which are set to 0.3 and 0.15 respectively.

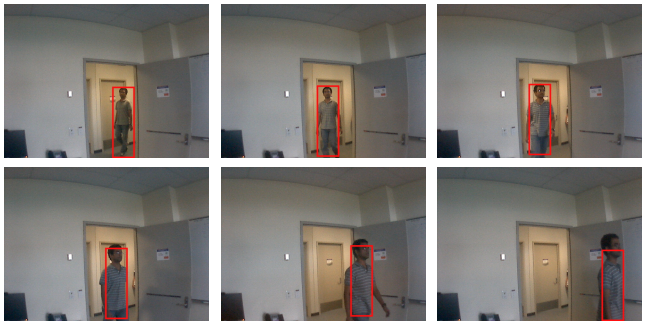
5. EXPERIMENTS

The performance of the proposed feature set in a framework of mean-shift tracking has been tested on a set of different video sequences. Fig. 3 shows the output of the algorithm on two video sequences. The first video sequence is recorded in NICTA's advanced surveillance group laboratory. In this video sequence, the target of interest is a walking person. As it is displayed in Fig. 3(i), the target of interest has very similar color as a background but with a different texture. Our experiments show that the proposed system outperforms in comparison to the color-based mean-shift tracker. The second video sequence is chosen from PETS2009 dataset. In this video sequence, the target of interest has a different color from the background which is suitable for color-based mean-shift tracking (Fig. 3(ii)). Our experiments show that in these cases, using local oriented feature set doesn't downgrade the performance of tracker. Furthermore in regards to robustness, due to invariance properties of the chosen feature set [15], the tracker performance is shown to be robust in presence of photometric variations, geometric transformations and noise.

6. PROPOSED ARCHITECTURE

6.1 Design Approach

The block diagram of the system is illustrated in Fig. 4. A hardware/software co-design approach is exploited in our system. While the mean-shift tracking block is implemented



(i) Test video sequence I

(ii) Test video sequence II

Figure 3: Test Video Sequences. The target of interest is a walking person which is shown in a bounding box. It is tracked from top left image to bottom right image.

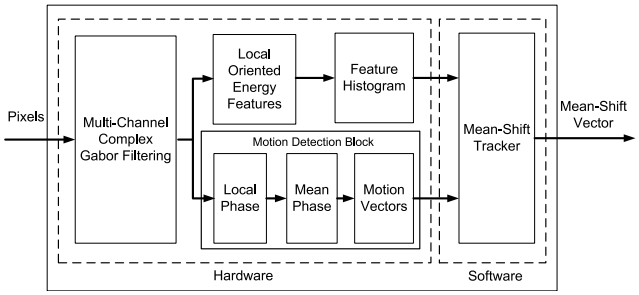


Figure 4: The block Diagram of the System

in software, the other blocks are implemented using hardware resources. This is mainly due to the fact that the mean-shift tracking algorithm is an iterative operation which can be easier implemented in software.

6.2 Multi-Channel Gabor Filtering

The Gabor filtering is implemented using hardware logic resources. Gabor filter is implemented using non-separable 2D convolution based architecture. The 2D convolution can be expressed as follows; Where the filter mask size is WxW, and $G(u, v)$ are the filter's coefficients at coordinates (u, v). The architecture of 2D convolution block is shown in Fig. 6.

$$W(x, y) = \sum_{u=-W}^{W} \sum_{v=-W}^{W} I(x + u, y + v)G(u, v) \tag{20}$$

The proposed system requires performing complex Gabor

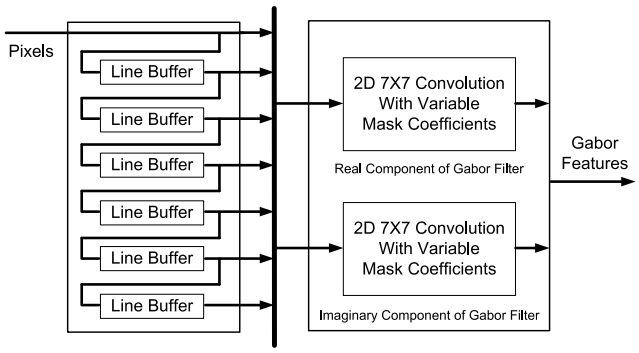


Figure 5: Multi-Channel Complex Gabor Filtering Block

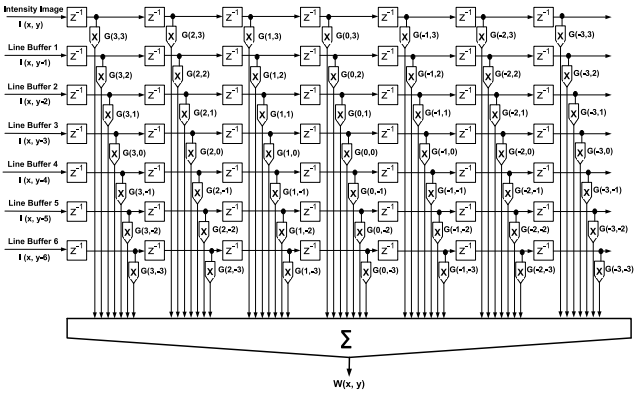


Figure 6: Architecture of 7x7 2D Convolution Block

filtering at 12 different channels. This requires 24 times 2D convolution blocks for computing the real and imaginary components of Gabor filter in total. Also the mask size of the filter is chosen to be 7x7 pixels. Convolving the incoming frames with 24 2D convolution blocks with the mask size of 7x7 pixels require extremely large amount of logic resources. To use the hardware logic resources efficiently, 2D convolution blocks with programmable coefficients are used. These blocks run at higher clock rates than the incoming data to perform multiple convolutions for the incoming images. The number of convolution which can be carried out using this technique is defined by the incoming pixel rate and the pixel rate which 2D convolutions can be executed. In our current implementation, video size of 640 X 480 pixels at frame rate of 30 fps with clock rate of 24 MHZ is used. Also the 2D convolution block can operate at maximum frequency of 296 MHZ on a Virtex-5 FPGA. Therefore with the aforementioned incoming and processing data frequency, we could perform 12 convolutions multiplexed in time using each convolution block. As it is shown in Fig. 5, the multi-channel complex Gabor filter blocks is composed of two convolution units, to compute the real and imaginary components of Gabor filter in 12 different channels. The mask coefficients of each channel are pre-computed and stored in a Read Only Memory (ROM) in twelve banks. Based on the number of a given filter, the coefficients of that specific filter are loaded into the convolution unit.

6.3 Local Features and Feature Histogram

The next stage of the system computes the local oriented features and feature histogram. The architecture of this block is shown in Fig. 7. Local oriented energy features are calculated in parallel in each channel for each pixel. The architecture of local phase and mean phase values computation block is shown in Fig. 8. The local features are temporarily buffered in the internal memory using FPGA's block RAMs and after certain number of cycles, these features are transferred to the external DDR2 memory.

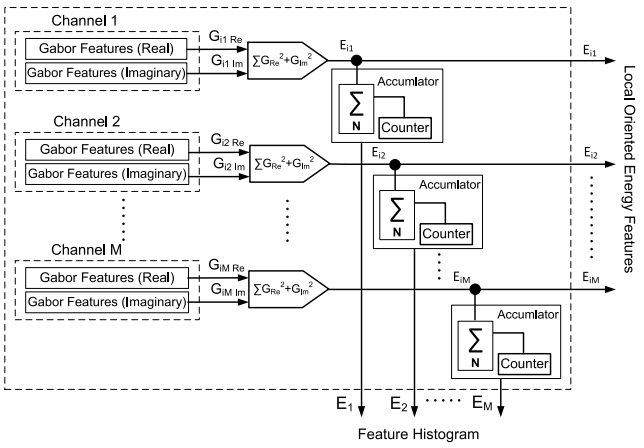


Figure 7: Architecture of Local Oriented Energy Features and Feature Histogram Computation block

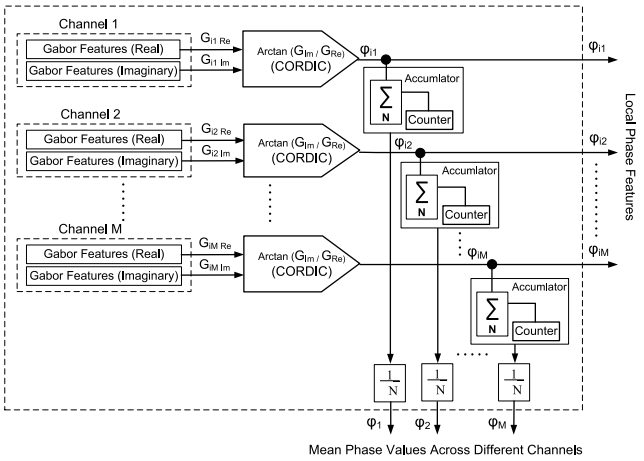


Figure 8: Architecture of Local and Mean Phase Computation block

7. IMPLEMENTATION RESULTS

The first implementation of the proposed algorithm was done in Matlab. Matlab was chosen as a developing environment because it allows fixed-point modeling of the system which is appropriate for hardware implementation. To achieve proof of concept, our first experiments were done using floating-point number representation. We use a 32-bit single precision floating-point number for this purpose. Once the desired goals were achieved, we developed a fixed-point version with variable bit-width and evaluated the performance of the system. It turned out that for bit-width higher than 13 bits, the output of the system was hardly distinguishable from floating-point version of the system.

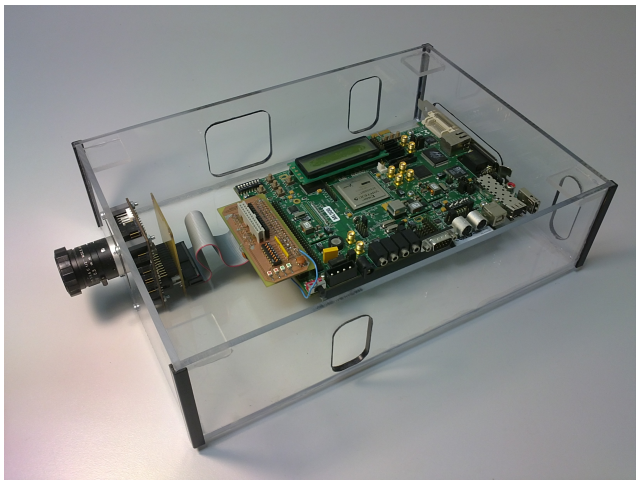


Figure 9: Xilinx ML506 Development Board with Micron MT9P001 Image Sensor

Hence, 16 bits was chosen as the bit-width of the fixed-point version.

The next stage was to design the system using hardware resources. Xilinx System Generator was used for this purpose. Different blocks of the system was developed using Xilinx system Generator blockset. Upon completion, the entire system was simulated using simlulink. To do this, the video file was converted to a bit stream using a frame to bit stream block. This mimics the data coming from an image sensor. Also the output of the system was changed to video format using a serial bit stream to frame buffer unit. Furthermore, we used hardware-in the loop co-simulation to reduce the simulation time.

Finally the entire system was implemented on Xilinx Virtex-5 XC5VSX50 using Xilinx ML506 development board (Fig. 9). The first three blocks of the system are implemented using hardware resources. The mean-shift tracking algorithm is implemented on Xilinx Microblaze soft core processor running at clock rate of 125 MHZ. The Microblaze was chosen for ease of use while in final version of the proposed system, a more powerful embedded processor can be used for implementation of mean-shift algorithm. The synthesis results are given in Tab. 1. For each building block of the system, the number of FPGA slices, lookup tables (LUTs), Flip Flops (FFs), Block RAMs (BRAMs) and Embedded Multipliers (MULTs) are given. Virtex-5 DSP48E blocks are used as the multipliers which can run at frequencies up to 550 MHZ. Also a Micron MT9P001 CMOS image sensor is used as the image sensor. The maximum resolution of the image sensor is 5 mega pixels at 14 FPS frame rate. In our experiments, the image sensor is configured to VGA (640 by 480 pixels) size and the pixel clock rate is set to 24 MHZ to adjust the frame rate at 30 FPS. The performance of the system was promising, achieving near real-time performance on image sizes of VGA and frame rate of 30 FPS. The performance of the system can significantly increase by using a more powerful embedded processor.

8. CONCLUSION

In this paper we proposed a real-time object tracking module for smart cameras. The main contribution of this paper was to propose a real-time object tracking system which uses feature sets other than traditional feature sets such as color to improve the accuracy and robustness of the system. To do this, we perform Gabor filtering across multiple channels on every incoming image. The multi-channel Gabor filtering stage can be considered as several 2D sheets of neurons, where each of them is tuned to a specific frequency, scale and orientation. By using this block, every incoming image is decomposed into different channels. By averaging the energies extracted from each channel, local oriented energy features are calculated. Local oriented energy features in different channels for every single pixel or block of pixels can be used as a much more discriminative feature set for higher level vision algorithms such as video object segmentation and tracking. In this paper, a mean-shift tracker was chosen to evaluate the performance of the proposed feature set. Our experiments show that it can significantly improve the robustness and performance of the algorithm. This becomes more important in cases that objects with similar color occlude each other or the objects have the same color as the background. Also due to invariance properties of the chosen feature set, it is shown to be robust in presence of photometric variations, geometric transformations and noise. Furthermore to improve the robustness, we estimated the motion pattern of the target region, using the local phase information. The motion pattern is used for displacement of the search area. To achieve real-time performance, the Gabor feature extraction part as well as local oriented energy feature, feature histogram and local phase computation parts are implemented on reconfigurable hardware while the mean-shift vector calculation is carried out in software. This is mainly due to nature of the operations required for the aforementioned tasks as they are sequential and iterative. Using hardware/software co-design approach, we could achieve near real-time performance on video size of 640 by 480 at frame rate of 30 FPS.

9. DISCUSSION AND FUTURE WORKS

In principle, a wide range of features can be chosen for representing the object for tracking. These include color, shape, texture and motion. In the proposed system architecture for object tracking on smart cameras, we have the color, motion and texture information for every pixel. One of our future works will be to integrate these feature sets and evaluate the performance of tracking. Another approach will be to use the on-line feature selection mechanism proposed by [3]. Adding this mechanism to the system, the feature sets is continuously evaluated over time so that the feature set which is better discriminating the objects from the background and object from one another will be chosen as the main feature set for tracking. With regards to speed of the algorithm, in our current implementation, mean-shift vector calculation is implemented on software. We are currently working on developing an efficient VLSI architecture for mean-shift vector calculation. We hope this can significantly reduce the computation time so that targets tracking can be done in real-time at higher resolution, higher frames with higher number of targets.

10. ACKNOWLEDGMENTS

This project is financially supported by the Australian Government through the National Security Science and Technology Branch within the Department of the Prime Minister and Cab-

Table 1: FPGA Implementation Results

Blocks	Slices	LUTs	FFs	BRAMs	MULTs
Total Availble Resources	8160	32640	32640	264	288
2D 7x7 Convolution Block	765	2398	2938	21	49
Multi-Channel Gabor Filter Block	1530	4796	5876	44	98
Local Oriented Energy Features Block	481	911	1899	29	24
Feature Histogram Computation Block	259	552	1023	13	12
Local and Mean Phase Computation Block	618	1126	1948	22	24

inet. This support does not represent and endorsement of the contents or conclusions of the project. NICTA is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

11. REFERENCES

[1] K. Cannons and R. Wildes. Spatiotemporal oriented energy features for visual tracking. *Asian Conference on Computer Vision, ACCV*, pages 532–543, 2007.

[2] J. Cho, S. Jin, X. Dai Pham, J. Jeon, J. Byun, and H. Kang. A real-time object tracking system using a particle filter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2822–2827, 2006.

[3] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[5] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[6] F. Dias, F. Berry, J. Serot, and F. Marmoiton. Hardware, Design and Implementation issues on a FPGA-based Smart Camera. In *ACM/IEEE International Conference on Distributed Smart Cameras*, pages 20–26, 2007.

[7] J. Díaz. *Multimodal bio-inspired vision system. High performance motion and stereo processing architecture.* PhD thesis, PhD dissertation, University of Granada, 2006.

[8] D. Fleet and A. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[9] D. Fleet and A. Jepson. Stability of phase information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1253–1268, 1993.

[10] D. Gabor. Theory of communication. Part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, 93(26):429–441, 1946.

[11] T. Gautama and M. Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 13(5):1127–1136, 2002.

[12] C. He, Y. Zheng, and S. Ahalt. Object tracking using the Gabor wavelet transform and the golden section algorithm. *IEEE transactions on multimedia*, 4(4):528–538, 2002.

[13] A. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern recognition*, 24(12):1167–1186, 1991.

[14] A. Jain, S. Prabhakar, L. Hong, and S. Pankanti. Filterbank-based fingerprint matching. *IEEE Transactions on Image Processing*, 9(5):846–859, 2000.

[15] J. Kamarainen, V. Kyrki, and H. Kalviainen. Invariance properties of Gabor filter-based features-overview and applications. *IEEE Transactions on image processing*, 15(5):1088–1099, 2006.

[16] T. Kubota. *Orientational Filters For Real–Time Computer Vision Problems*. PhD thesis, Georgia Institute of Technology, 1995.

[17] C. Liu and H. Wechsler. Independent component analysis of Gabor features for face recognition. *IEEE Transactions on Neural Networks*, 14(4):919–928, 2003.

[18] L. Ma, Y. Wang, and T. Tan. Iris recognition based on multichannel Gabor filtering. In *Proc. Fifth Asian Conference on Computer Vision, ACCV*, volume 1, pages 279–283. Citeseer, 2002.

[19] F. Mühlbauer and C. Bobda. A dynamic reconfigurable hardware/software architecture for object tracking in video streams. *EURASIP Journal on Embedded Systems*, 2006(1):16, 2006.

[20] J. 'Schlessman, C. Chen, W. Wolf, B. Ozer, K. Fujino, and K. Itoh. Hardware/software co-design of an fpga-based embedded tracking system. pages 123–123, 2006.

[21] M. Tomasi, J. Díaz, and E. Ros. Real Time Architectures for Moving-Objects Tracking. In *Proc. Reconfigurable computing: architectures, tools, and applications: third international workshop*, pages 365–372. Springer-Verlag New York Inc, 2007.

[22] K. Yamaoka, T. Morimoto, H. Adachi, T. Koide, and H. Mattausch. Image segmentation and pattern matching based FPGA/ASIC implementation architecture of real-time object tracking. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 181–186. IEEE Press, 2006.