Real Time Motion Analysis Toward Semantic Understanding of Video Content

Yong Wang[1], Tong Zhang, Daniel Tretter
Imaging Systems Laboratory
HP Laboratories Palo Alto

HPL-2005-93
May 16, 2005*

real time video motion analysis, video semantics, video understanding, camera motion estimation, moving object detection, moving object tracking

Video motion analysis and its applications have been a classic research topic for decades. In this technical report, we explore the problem of real time video semantics understanding based on motion information. The work can be divided into two segments: global/camera motion estimation and object motion analysis. The former involves optical flow analysis and semantic meaning parsing, and the latter involves object detection and tracking. Although each of these topics has been studied extensively in the literature, a thorough system combining all of them without human intervention, especially under a real time application scenario, is still worthy of further investigation. In this paper we develop our approach toward such a destination and propose an integral architecture. The usability and efficiency of the proposed system have been demonstrated through experiments. Results of this project have numerous applications in digital entertainment, such as video and image summarization, annotation, retrieval and editing.

* Internal Accession Date Only
Presented at and published in the Conference on Visual Communications and Image Processing, 12-15 July 2005, Beijing, China.

[1] Yong Wang is currently a graduate student at Columbia University. Presented in this report is his summer intern work at HP Labs

Real Time Motion Analysis Toward Semantic Understanding of Video Content

Yong Wang, Tong Zhang, Daniel Tretter

1. INTRODUCTION

The work presented in this technical report was initially inspired by the application scenario of video printing. Current printers are good at printing planar media content such as documents and images, but not for 3-D signals such as video, which contains much more information with a huge amount of redundancy. One intuitive way to print video is to select key frames from the clip. In previous work we proposed an intelligent method for extracting key frames by analyzing audio and video features[1]. In this report, we will focus on the video motion analysis part of this method.

Video motion analysis can be divided into two aspects: global / camera motion estimation, and foreground / object motion analysis. Camera motion estimation involves optical flow analysis, camera motion estimation and semantic meaning extraction; object motion analysis involves object detection and tracking. Although these topics have been extensively discussed in the literature, a thorough system combining all of them without human interaction, especially under a real time application scenario, is still worthy of further investigation. We specify our approach toward such a destination and propose an integral architecture. Based on this architecture we also build up a real time video semantics analysis tool and demonstrate its usability and efficiency through experiments. The result of this project can be widely used in video / image understanding and management applications such as summarization, annotation and retrieval.

This report is organized as follows. Section 2 presents an overview of the proposed system architecture for real time video motion analysis. Section 3 discusses issues in camera motion estimation. Section 4 describes the method for moving object detection and tracking. Experimental results are shown in section 5. Finally, we present our conclusions in section 6.

2. SYSTEM ARCHITECTURE

Figure 1 outlines the framework of the proposed system for real time video motion analysis. It can be divided into four modules: data preparation, camera motion understanding, object detection and object tracking.

The proposed system is built upon pixel-domain processing. The data preparation module provides the necessary image sequence for the subsequent processing. Specifically, after the video bit stream is decoded, spatial down-sampling is applied to reduce the computational complexity to fulfill the real time requirement. The low-resolution image is used later in the object motion analysis module, and the luminance channel is used for the camera motion analysis module.

The camera motion understanding module is comprised of camera motion estimation and subsequent semantic meaning parsing. The former is implemented through optical flow analysis and affine model estimation. The featured block selection component picks image blocks with significant detail for camera motion estimation. The semantic meaning parsing component tries to understand the semantic intention behind the camera motion activities. We propose a simple but effective summarization method to achieve this purpose, which will be specified in Section 3.

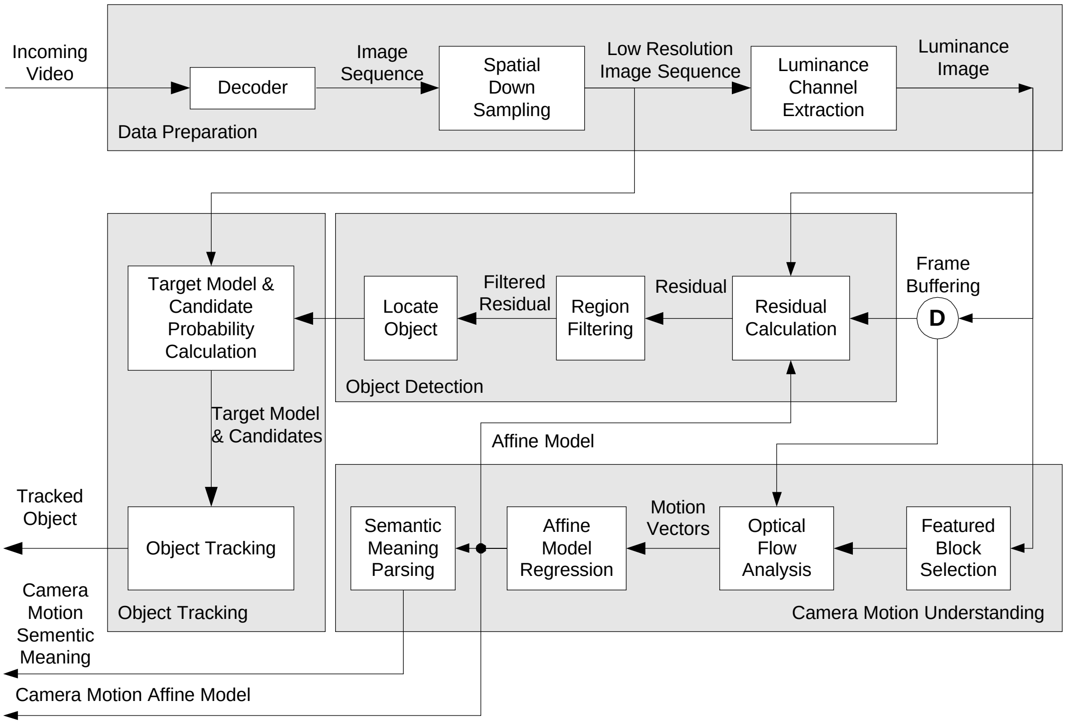


Figure 1: Proposed system framework for real time video motion analysis.

The object detection module locates semantically meaningful objects through motion residual error analysis. First the residual error is estimated using affine model parameters, as well as the current and the previous (buffered) frames. The residual errors, however, cannot be directly used to locate the object due to interference such as signal noise, spatial texture complexity and affine model error. Therefore, region filtering is employed to exclude outliers. The basic philosophy of the filtering is to find the dominant object with remarkable motion, moderate size and semantically meaningful position. This procedure is implemented through a set of filters, as specified in Section 4. The region boundary of the detected moving object is then fed into the object tracking module.

The object tracking module follows the trajectory of the detected moving object. RGB frames, instead of luminance images, are used during this procedure, allowing us to apply color information to facilitate the tracking. Candidate regions within the searching range around the detected object are checked through some similarity estimation; the optimal one is selected; and the trajectory is recorded accordingly.

3. CAMERA MOTION ESTIMATION

Camera (global) motion refers to the motion induced by camera operations such as zooming, panning and rotation. Its effect can be observed as frame-wide well-regulated optical flow changes following the affine transformation defined by the camera motion. In real video this ideal situation is somewhat impaired by the foreground motion and other signal noises. The task here is to discover the camera motion's affine model based on observed noisy optical flow activities. We also discuss the issue of high level semantic meaning extraction based on the estimated camera motion parameters.

3.1. Optical Flow Analysis

The basic method for motion estimation is the block matching algorithm (BMA). The matching criterion calculates the intensity difference between a block at position (*m*, *n*) with dimension (*W*, *H*) in the k^{th} frame and a block shifted by the motion vector (*i*, *j*) in the $(k\text{-}1)^{th}$ frame, as:

$$E_{k,m,n}(i,j) = \sum_{x_1=m}^{W+m-1} \sum_{x_2=n}^{H+n-1} e(I_k(x_1,x_2),\ I_{k-1}(x_1+i,x_2+j)) \qquad (1)$$

where *I* is the luminance intensity of the pixel, and *e*(*i*, *j*) is the error metric, usually sum of the squared error or sum of the absolute error. The motion vector is the one yielding minimum distortion within a defined search range (*M*, *N*), i.e.:

$$V(k,m,n) = \underset{i,j}{\arg\min}\left\{ E_{k,m,n}(i,j) \middle| 0 \le |i| \le M, 0 \le |j| \le N \right\} \qquad (2)$$

This minimum distortion measurement, however, doesn't always deliver the right information, as illustrated in Figure 2(a). At the moment of this frame, the camera was undergoing a panning operation with a direction indicated by the lines from the center of the blocks, except for the one with a cross mark, which has a quite different motion vector compared with the others. The reason for this discrepancy is that the distortion distribution among the search range (in this case M=N=8) has a very small variance and the final result is affected by random noise.

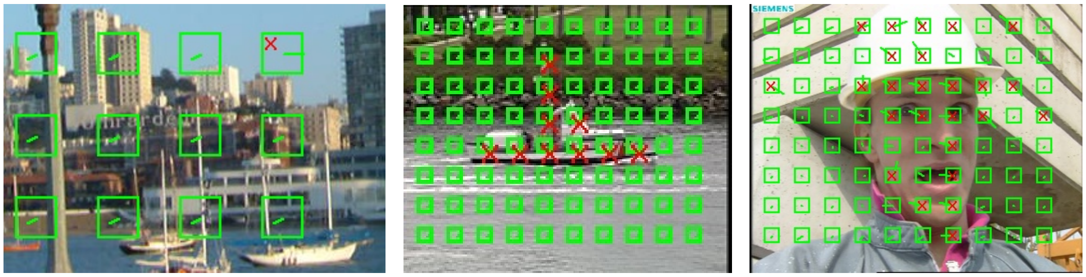
3.1.1. Outlier analysis

In this paper, we define outliers as blocks with motion vectors different from the consensus obtained from all blocks in the frame. Generally most outliers fall into two categories:

1) Object motion: these outliers carry useful information about object motion, as indicated in Figure 2(b), where in the video sequence, the boat has its own motion different from the camera motion. We will discuss utilizing outliers to detect the object motion in Section 4.
2) Mismatched block: this happens for several reasons. In Figure 2(c), blocks located on the building's wall have very simple region textures, and consequently yield too small distortion distribution variance in the search range. Other reasons include sudden luminance change and limited search range compared with the camera motion magnitude.

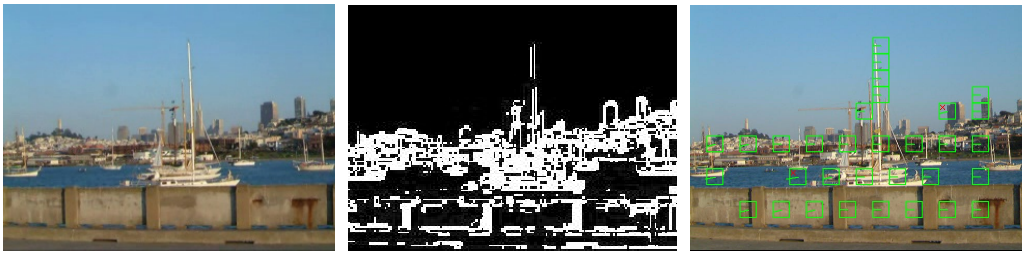
3.1.2. Featured block selection

Featured block selection is used to choose specific blocks with trackable content. It is important in two senses: first, it can efficiently reduce the amount of outliers as mentioned above; second, it can reduce the BMA calculation significantly, which is required for a real time application scenario. There are several feature selection approaches in the literature, such as KLT transform, temporal texture analysis, DCT energy statistic, Moravec operator, etc. In this work, we use edge based block selection considering its potential advantage in object detection. After edge detection, only blocks containing a certain amount of edge pixels are selected and used for motion compensation. The decision threshold is dynamically adjusted according to the statistical distribution of the image's gradient magnitude. As illustrated in Figure 3, through feature selection, the amount of outliers due to mismatched blocks is effectively reduced.



(a) (b) (c)

Figure 2: Motion estimation outliers due to object motion and noise.



(a) Original frame (b) Edge map (c) Selected featured blocks

Figure 3: Edge based feature selection.

3.1.3. Fast motion estimation

In order to find the motion vector defined by Equation (1), an exhaustive search within the search range is the simplest way and guarantees a globally optimal solution. However, its computational cost is too high for real time applications. Therefore fast searching approaches[2] in the literature are considered to speed up the procedure. In addition, the following measures are also taken to further reduce the computation:

1) Image resolution down sampling. The motion compensation is executed at a low-resolution image level. Our experiments showed that compensation at a low-resolution level can achieve satisfatory results in terms of semantic meaning extraction and object detection, while saving computational cost.
2) Block skipping. If one block is selected, all of its neighboring blocks are skipped under the assumption that adjacent blocks share very similar motion behavior. This block sampling can be seen in Figure 2.
3) Halfway termination. In calculating the compensation residual error of a certain block, once the accumulated distortion during summing up the pixel differences is bigger than the current minimum of block distortion, the calculation is terminated and this block is skipped. This measure helps to avoid unnecessary calculation..

3.2. Affine Model Estimation

The affine model is widely used in estimating camera motion[3]. In this work we consider rotation, panning and zooming of a camera. That is, an arbitrary point P(x, y, z) in the space is projected into the camera's view plane as the point Q(u, v) following the transform:

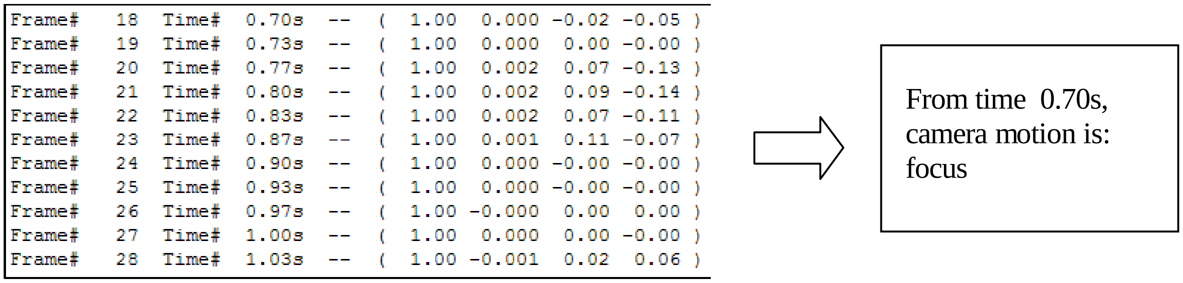
$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} zoom & rotation & pan_x \\ rotation & zoom & pan_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3}$$

where *zoom*, *rotation*, pan_x and pan_y are four parameters determined by the camera motion. Since there is no depth mapping information for non-stereoscopic video signal, we assume a constant depth for all points, i.e. z=1. This model satisfies our requirement of semantic meaning analysis for a wide range of videos. The affine model is estimated by using the least squared error (LSE) regression[4].

The basic idea of affine model iteration is to exclude the outliers whose residual errors are more than a pre-defined threshold. The threshold is usually predetermined based on the standard deviation of the residual error. After the outliers are excluded, the affine model is re-estimated. The iteration is terminated when the affine parameter set becomes stable. To avoid potential divergence, a maximum iteration amount is defined. Our experiments indicated that the estimation is done within three iterations for most frames.

3.3. Semantic Meaning Extraction

After affine model estimation, a set of parameters is obtained for each frame denoting the camera motion at that moment. However, to retrieve motion semantics, it is necessary to have a summarization based on these models over moderate time spans. Figure 4 shows a series of affine model parameters, denoted as (*Zoom*, *Rotation*, Pan_x Pan_y), in their frame order. This data, while complete, is not in a human-friendly format. A more suitable way to summarize the results is to generate semantic level descriptions such as "from time 0.7s, the camera is focusing without motion". In this paper, we propose the following technique for abstracting camera motion.



```
Frame#   18  Time#  0.70s  --  (  1.00  0.000 -0.02 -0.05 )
Frame#   19  Time#  0.73s  --  (  1.00  0.000  0.00 -0.00 )
Frame#   20  Time#  0.77s  --  (  1.00  0.002  0.07 -0.13 )
Frame#   21  Time#  0.80s  --  (  1.00  0.002  0.09 -0.14 )
Frame#   22  Time#  0.83s  --  (  1.00  0.002  0.07 -0.11 )
Frame#   23  Time#  0.87s  --  (  1.00  0.001  0.11 -0.07 )
Frame#   24  Time#  0.90s  --  (  1.00  0.000 -0.00 -0.00 )
Frame#   25  Time#  0.93s  --  (  1.00  0.000 -0.00 -0.00 )
Frame#   26  Time#  0.97s  --  (  1.00 -0.000  0.00  0.00 )
Frame#   27  Time#  1.00s  --  (  1.00  0.000  0.00 -0.00 )
Frame#   28  Time#  1.03s  --  (  1.00 -0.001  0.02  0.06 )
```

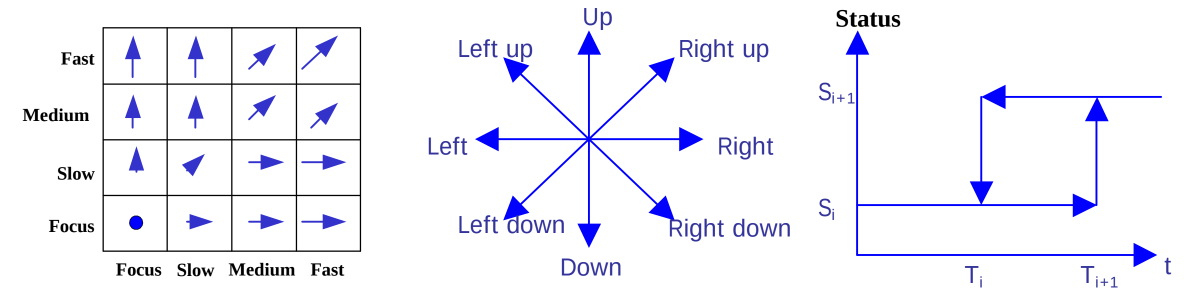
From time 0.70s,
camera motion is:
focus

Figure 4: Semantic meaning abstraction from affine model parameters.

1) Camera motion quantization. The camera motion is quantized into several high level conceptions. Figure 5 is the schema employed to quantize the camera pan parameters. Other parameters can be processed similarly. In this schema, the pan magnitude along X-, Y- axes are classified into four regions: focus, slow, medium and fast. Accordingly, the direction is also

quantized into discrete values as indicated in Figure 5(b). In order to avoid interference due to camera vibration and noise, hysteresis thresholding is employed to decide the transitions.

2) Sentence generation. Small time spans are naturally clustered together based on their affine model similarity, yielding a reasonable number of semantic meaning regions. That is, any adjacent frames sharing the same quantized camera motion behavior are merged together as one region, which we call a sentence. This step segments the video into a number of sentences, and within each sentence the camera motion is consistent. The average affine model is calculated for each sentence.
3) Paragraph generation. Next, the histogram of sentence duration is calculated. A further merging operation is executed. That is, consecutive sentences can be merged into one paragraph according to a similarity comparison. The similarity is measured according to some distance metrics between affine models. The semantic meaning of each paragraph is re-evaluated using a weighted affine model. This procedure is illustrated in
4) Figure 6.



(a) Quantized magnitude space (b) Quantized motion direction (c) Hysteresis thresholding

Figure 5: Camera motion quantization.

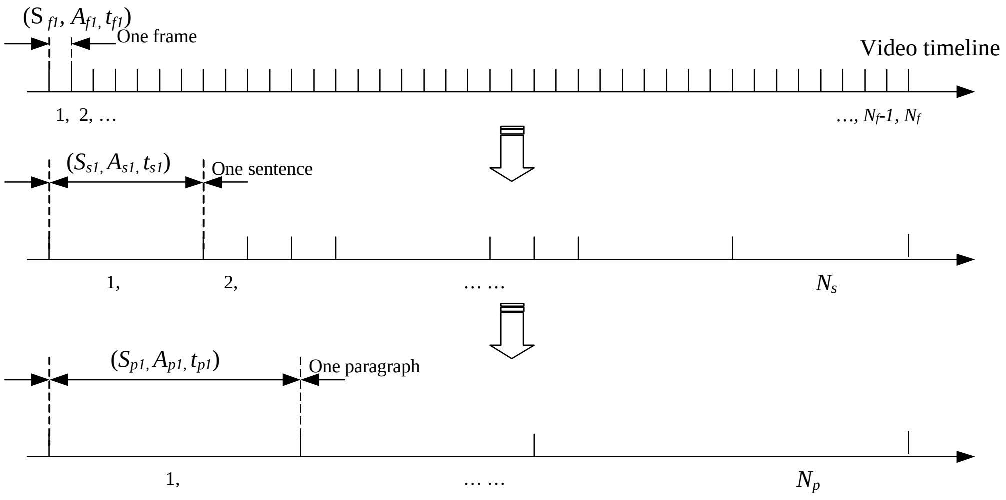


Figure 6: Similarity based clustering in semantic meaning abstraction.

Suppose there are N_f frames in the entire video clip. For each frame i, i=1, 2, …, N_f, there is the duration t_{fi}, affine model A_{fi}, and quantized camera motion S_{fi}. After sentence generation, N_s sentences are produced, each with average affine model A_{sj} and duration t_{sj}, where

$$\begin{aligned} A_{sj} &= \frac{\sum_i A_{fi} \cdot \left\{1 \middle| S_{f_i} = S_{f_{(i+1)}} \right\}}{\sum_i \left\{1 \middle| S_{f_i} = S_{f_{(i+1)}} \right\}} \quad j = 1, 2, ..., N_s \\ t_{sj} &= \sum_i t_{fi} \cdot \left\{1 \middle| S_{f_i} = S_{f_{(i+1)}} \right\} \end{aligned} \tag{4}$$

Then, the histogram statistics are calculated for the distribution of duration t_{sj}. A pre-defined duration tolerance threshold T_d is used to merge sentences into paragraphs. The sentences with duration larger than T_d will act as anchors S_a, and the other sentences are merged into these anchors based on distance measurement. The semantic meaning of each paragraph is re-evaluated as:

$$\begin{aligned} A_{pk} &= \frac{\sum_{t_{s_a,k} < t_{s_j} < t_{s_a,k+1}} A_{s_j} \cdot t_{s_j} \cdot M(A_{s_j}, A_{s_a,k})}{\sum_j t_{s_j} \cdot M(A_{s_j}, A_{s_a,k})} \quad k = 1,\ 2,\ ...,\ N_p \\ M(A_s, A_{s_a,k}) &= \left\{1 \middle| D(A_s, A_{s_a,k+1}) > D(A_s, A_{s_a,k}) \right\} \\ S_{pk} &= Q(A_{pk}) \\ t_{pk} &= \sum_j t_{s_j} \cdot M(A_{s_j}, A_{s_a,k}) \end{aligned} \tag{5}$$

where $Q(A)$ is the quantization mapping from affine model to semantic meaning, and D is the selected distance measurement. In this process, T_d embodies the sensitivity degree in terms of semantic meaning change detection. How to select T_d to achieve the appropriate tradeoff between particularity and generality is worthy of further investigation.

4. MOVING OBJECT DETECTION AND TRACKING

4.1. Moving Object Detection

Object detection attempts to find and track a single dominant object region with considerable motion and semantic meaning. In general, object detection is a difficult problem and approaches in the literature usually involve some degree of simplification. Some examples of such simplification include limiting the application to specific object category such as face[5], and using pre-known models[6]. In this paper, we detect moving object only based on motion information, with the assumption that the foreground motion behavior is different from the background and its region is exposed as outliers when estimating motion compensation residual errors as in Equation (1). In reality, this assumption is not always true due to noisy background motion estimation. Also in a real time system, computing simplifications make the situation worse. Therefore, one important task for motion based object detection is to distinguish the actual object from the background noise among the outliers. In order to achieve this, we make some simplification assumptions:

1) Moderate object size assumption: the object should be of moderate size in terms of exposing enough residual errors and delivering considerable semantic meaning. Too small an object is difficult to detect due to its limited residual information and the interference from noise, while too large an object will make the performance of camera motion estimation very poor and consequently the judgment of object is also unreliable.
2) Center-biased assumption: the interest in an object wanes as its location moves away from the center of the frame. This assumption is reasonable because it matches the subjective experience well. Imagine a sequence with a car driving through the screen, the frames with the car close to the middle of the scene are likely to be of more interest than the frames with the car entering or leaving the scene.

While there are tradeoffs in making such simplifications, our experiments indicate that these assumptions match the semantic meaning criteria well and are reasonable and efficient simplifications.

With the above assumptions, instead of using the directly calculated residual error for detecting moving objects, we apply a set of weighting parameters to filter the residual errors. Figure 7 shows the diagram of this filtering. The residual error of each incoming video frame is estimated after the camera affine model is obtained. Then the error undergoes a set of filters and the object region is detected after the filtering. These filters are detailed below.

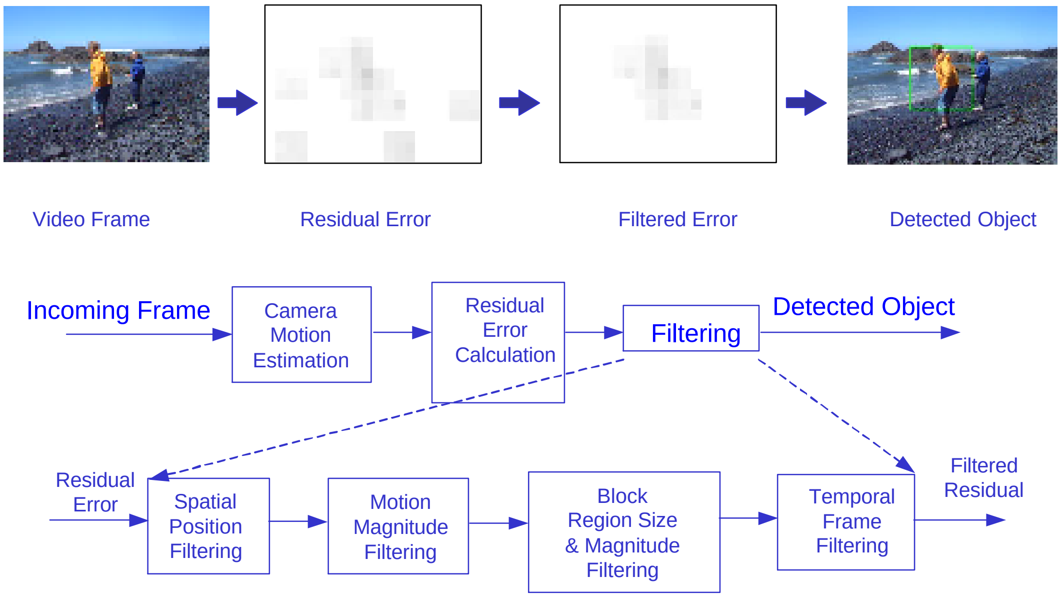


Figure 7: Moving object detection based on filtered residual error.

1) Spatial position filtering: this filter assigns higher weight to blocks located near the central region of the frame. One possible filter function is an exponential function:

$$f_s(P) = \frac{1}{1-\varepsilon} \cdot \left(1 - e^{-\lambda(M - \|P\|)}\right), \quad \lambda = \frac{-\log \varepsilon}{M} \qquad (6)$$

where P is the block under estimation. $\|P\|$ is the distance between this block and the center of the frame. M is the predefined value where the weight is zero. e is a parameter used to normalize the function.

2) Motion magnitude filtering: this filter comes from the empirical observation that the bigger the camera motion's magnitude is, the more outliers are prone to be generated by background noises. Currently the panning magnitude is considered. One possible filter function is Gaussian based:

$$f_m(\|P_M\|) = e^{\frac{-\|P_M\|^2}{2s^2}} \qquad \|P_M\| = \sqrt{pan_x^2 + pan_y^2} \tag{7}$$

where $\|P_M\|$ is the panning magnitude, and the deviation is set to be on the boundary of the searching range. The difference between this and the previous filter is their behavior on the boundary: the spatial position filter assigns zero weight on the frame boundary while the motion magnitude filter has non-zero weight on the search boundary.

3) Block region size and magnitude filtering: in order to realize this filter, all outlier blocks are firstly clustered into regions based on their connectivity. This is achieved by using a fast flooding algorithm. Once all of the outlier blocks are clustered into specific regions, each region is then denoted by a bounding rectangle box and a representative residual magnitude, which is selected to be the maximum residual error among all of the blocks in the region. Next, the size and the representative residual magnitude are estimated. The region with a size below some threshold, or magnitude under some threshold, will be ignored. If there is at least one region with a size larger than a third threshold, this frame is skipped and will not be used to detect object because we assume the affine model in this frame is not reliable. All of the thresholds used here are set in advance.

4) Temporal frame filtering: this filter is necessary because not every frame is suitable for detecting moving object. Figure 8 shows one frame and its residual error map. Since the residual error of the moving object (the player) is overwhelmed by the background noise, it is very hard to locate the object. The residual error from the audience is so large due to its complex texture and the very fast motion of the camera. Thus, after all the above filtering, only those frames with one dominant residual block region and moderate size will be kept. All other frames are skipped.

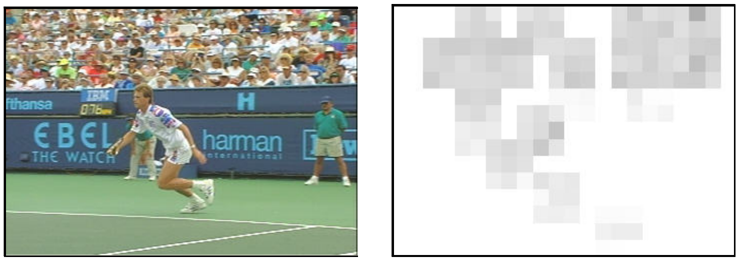


Figure 8: Not every frame is suitable for detecting moving object.

Once a moving object is detected, its position and size (denoted by a bounding rectangle box) is forwarded to the moving object tracking module.

4.2. Moving Object Tracking

In this work, moving object tracking is based on the algorithm of kernel based tracking (KBT)[7, 8].

4.2.1. Feature space representation

The quantized color histogram is used to represent the detected object during object tracking. We directly use RGB color space with $N \times N \times N$ bins, where N is the number of bins for each color channel. The advantage of color histogram is its robustness when the object is undergoing complex motion such as rotation, scaling, and even non-rigid warping. However, the drawback of color histogram is its lack of spatial or texture information. In this work, to compensate this problem, we propose a spatially enhanced color histogram. In one implementation, the object region is divided into the marginal part and the central part, as shown in Figure 9. Each part has its own histogram statistics, and the overall histogram is the concatenation of the two regional histograms. Please note that other ways of partitioning the color histogram with spatial information could be used as well.

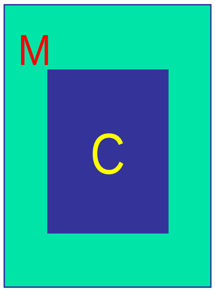


Figure 9: Spatial information enhanced color histogram.

The color histogram is normalized to a probability shape:

$$q_u = \frac{\sum_{i=1}^{n} \delta[b(x_i) - u]}{n} \tag{8}$$

where δ is the Kronecker delta function, u is the histogram entrance index, and q is the color histogram probability. $b(x_i)$ is the function mapping a pixel x_i, located in the object region, into one entrance of the histogram bins. n is the number of pixels in the object region.

The direct usage of color histogram has a drawback: for the peripheral pixels, they are prone to be affected by occlusions (clutter) or interference from the background. The improvement to this situation is to apply a kernel function and assign smaller weights to pixels farther away from the center. Using these weights can effectively increase the robustness of the density estimation.

4.2.2. Kernel based object tracking

The principle of kernel selection is to select one with a convex and monotonic decreasing profile. The recommended kernel function is the Epanechnikov kernel:

$$k(x) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2)(1-x) & if\ x \le 1 \\ 0 & otherwise \end{cases} \tag{9}$$

where c_d is the volume of the unit d-dimensional sphere and x is the normalized position of the pixels in the object region. The origin of the normalized coordinates is located at the center of the object. The Epanechnikov kernel has a constant derivative, which yields significant computing simplification. Correspondingly the target model is defined as:

$$\hat{q}_u = C \sum_{i=1}^{n} k\left(\left\| x_i^* \right\|^2 \right) \delta \left[b\left(x_i^* \right) - u \right] \tag{10}$$

where C is a normalization constant. So the target model is a weighted color histogram probability distribution function. Similarly the target candidate is defined as:

$$\hat{p}_u(y) = C_h \sum_{i=1}^{n_h} k\left(\left\| \frac{y - x_i}{h} \right\|^2 \right) \delta \left[b(x_i) - u \right] \tag{11}$$

where C_h is the normalization constant, h is the bandwidth parameter to handle the object scaling, and y is the central coordination of the target candidate.

Given the target model and the target candidate, the similarity metric is measured by using the Bhattacharyya coefficient, which is defined as:

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

where m is the bin number of the histogram. Further, the distance between the target model and the candidate is defined based on the Bhattacharyya coefficient:

$$d(y) = \sqrt{1 - \rho\left[\hat{p}(y), \hat{q} \right]} \tag{13}$$

Thus, the object-tracking task finds the target candidate that minimizes this distance.

5. EXPERIMENTAL RESULTS

The proposed system has been validated through extensive simulation experiments. In the simulation, we adopted the Sobel operator to extract feature blocks, the three-step searching algorithm to run the optical flow analysis, the LSE regression iteration to work out the affine model, and the Bhattacharyya coefficient to estimate the object region similarity. Two major aspects of the system are evaluated: computational complexity for real time processing and motion analysis performance. The simulation was run on a laptop with configuration of PIII 1.0GHz CPU and 256M memory.

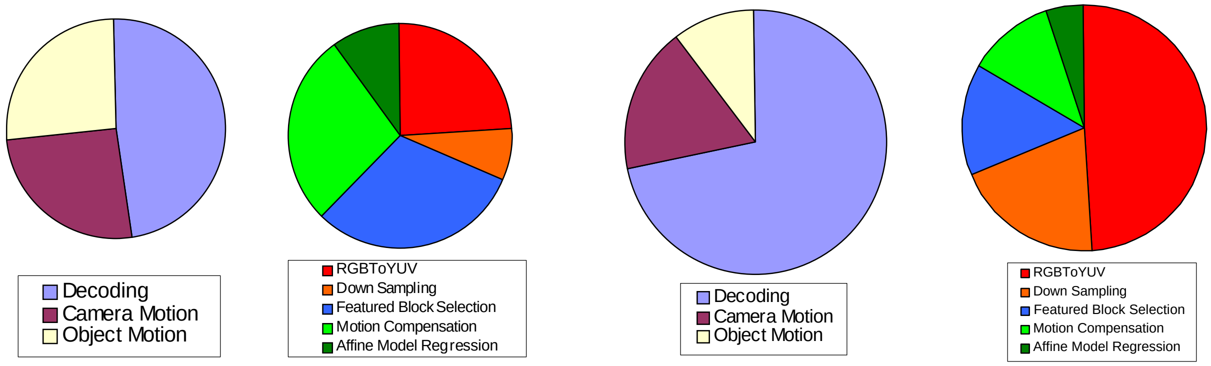
5.1. Computational Complexity

Table 1 summarizes the result for the video clip *Stefen*. The video format is detailed in the table. From it we can see that the real time performance of our proposed method is very promising: the total time of motion analysis is comparable to the decoding cost and the whole processing including decoding, camera motion analysis and object motion analysis (detection and tracking) can be finished several times faster than real time. A zoomed observation in Figure 10 gives us some more information. Figure 10(a) shows the time spent on decoding, camera motion and object motion analysis. If we analyze the image sequence instead of the encoded video clip, the decoding cost can be saved and the procedure can be sped up by about two times. Figure 10(b) details the cost of camera motion analysis and is helpful for us to further improve the performance by optimizing dominant components. For example, if we directly work on the luminance image, the cost of color space transform can be saved. Also, the featured block selection part (mainly edge detection) consumes about one third of the cost in camera motion estimation. A faster featured block selection method may yield better performance.

Table 1: Real-time performance of the system when applying to the *Stefan* video

Sequence	Coding Format	Image Format	FPS	Length (s)	Decoding Time (ms)	Decoding + Camera Motion (ms)	Decoding + Camera Motion + Object Motion (ms)
Stefan	MPEG-1	SIF (352 X 240)	30	10.0	2016	3105	4256

Table 2 shows the result for another video clip *San Francisco*, which is a panorama view of a wharf in San Francisco. We can get similar conclusion on this sequence, which means our method has a very good practicability for typical types of video. Figure 10(c) and (d) detail the time spent by each component. Since the image resolution is relatively high, the cost of decoding, color space transform and down sampling dominate the total time spent.



(a) *Stefan* (b) *Stefan* (c) *San Francisco* (d) *San Francisco*

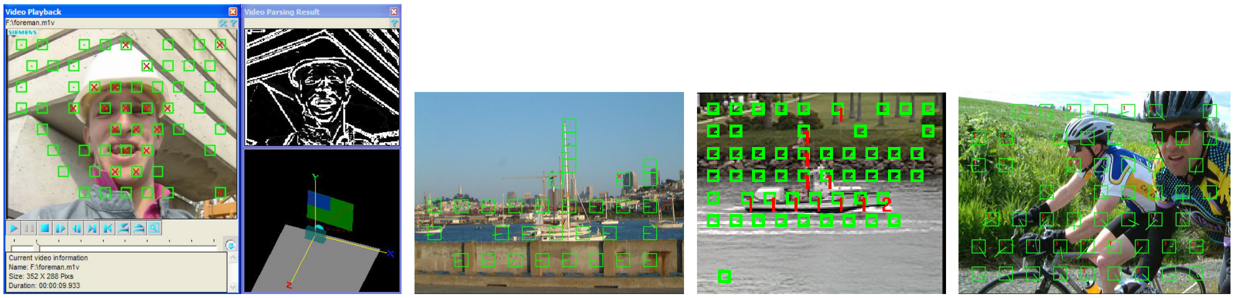
Figure 10: Computational cost details for two video clips: *Stefan* and *San Francisco*.

Table 2: Real-time performance of the system when applying to the *San Francisco* video

Sequence	Coding Format	Image Format	FPS	Length (s)	Decoding Time (ms)	Decoding + Camera Motion (ms)	Decoding + Camera Motion + Object Motion (ms)
San Francisco	MJPEG	(640X480)	15	26.0	9263	11636	12959

5.2. Camera Motion Estimation

Figure 11(a) shows the user interface of camera motion estimation. The visual output includes the edge map, selected feature blocks with motion vector, detected outliers and a simulated 3D camera motion. In general, the selected feature blocks can provide sufficient information for optical flow analysis, and the LSE regression can detect outliers effectively. Subjective tests indicate that the simulated camera motion, which is based on the estimated affine model, coincided with human perceptual observation. More results are shown in Figure 11 (b)-(d). Figure 11(b) and (c) are examples where the camera motion analysis worked well. While Figure 11(d) shows a case where the estimation failed due to two reasons: firstly the video was taken by a bicycle rider therefore the camera was undergoing a rapid translation operation, which is not modeled by our current four-parameter affine model; secondly the scene is dominated by the foreground objects (two riders) with ample non-rigid motion. As mentioned earlier, for such a sequence there is not enough statistical data about the background motion and the estimated affine model is unreliable.



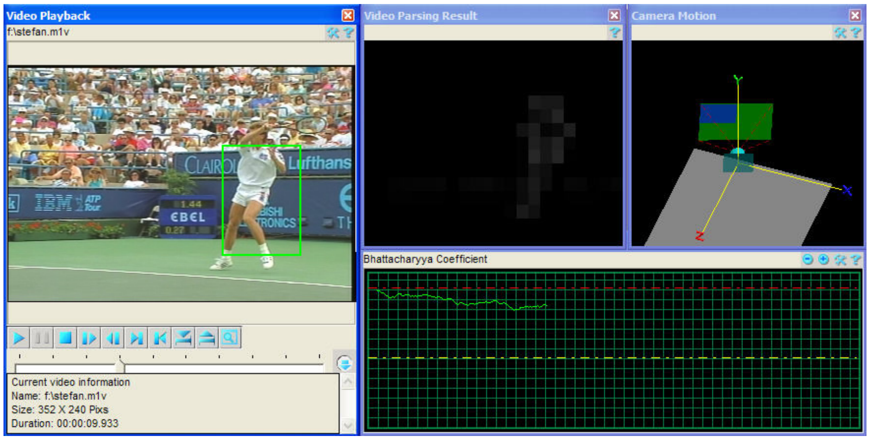
Current video information
Name: F:\foreman.m1v
Size: 352 X 288 Pixs
Duration: 00:00:09.933

(a) User interface (b) San Francisco (c) Coast guard (d) Bicycle

Figure 11: Results of camera motion estimation.

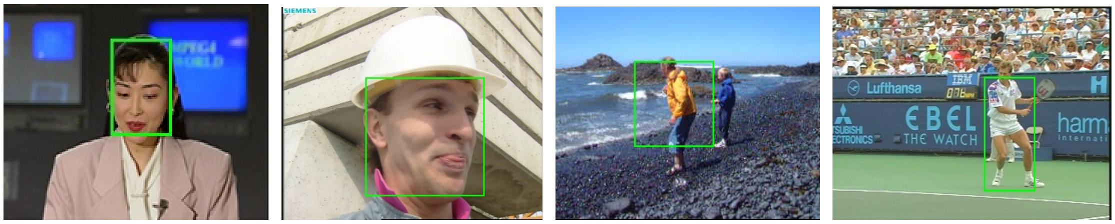
5.3. Object Motion Analysis

Object motion analysis includes moving object detection and tracking. Figure 12 shows the user interface, including the simulated camera motion, the residual error map, the detected / tracked moving object, and the Bhattacharyya coefficient value for the tracked object region. Figure 13 illustrates more moving object detection results, together with the frame index where the object was detected.



Current video information
Name: f:\stefan.m1v
Size: 352 X 240 Pixs
Duration: 00:00:09.933

Figure 12: User interface of moving object detection / tracking.



(a) Akiyo Frame# 80 (b) Foreman Frame# 7 (c) Seaside Frame# 18 (d) Stefan Frame# 3

Figure 13: Example results of object motion detection.

6. CONCLUSION

In this work, we explored the research topic of real time video motion analysis. We discussed two aspects of our system: camera motion estimation and object motion analysis, with the latter including moving object detection and tracking. We proposed an integral architecture and built up a real time video motion analysis system. The usability and efficiency of the proposed system were demonstrated through experiments. It was shown that the computational complexity and the analysis performance are well balanced in the system. Valuable benchmark data were provided for further improvements. The result of this project can be widely used in video understanding and management applications such as browsing, indexing, printing and summarization of video.

REFERENCES

1. T. Zhang, “Intelligent keyframe extraction for video printing,” *Proc. of SPIE’s Conference on Internet Multimedia Management Systems*, vol. 5601, pp.25-35, Philadelphia, Oct. 2004.
2. R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation,” *IEEE Trans. On Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438-42, Aug. 1994.
3. J.-II Park, N. Yagi, K. Enami, *et al.*, “Estimation of camera parameters from image sequence for model based video coding,” *IEEE Trans. On Circuit and System for Video Technology*, vol. 4, no. 3, June 1994.
4. R. L. Rardin, *Optimization in Operations Research*, Prentice Hall, 1998, ISBN: 0023984155.
5. H. A. Rowley, S. Baluja, and Takeo Kanade, “Neural network-based face detection,” *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, January 1998.
6. C. Schlosser, J. Reitberger, and S. Hinz, “Automatic car detection in high resolution urban scenes based on an adaptive 3D-model,” *Proc. of IEEE/ISPRS joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, pp. 167-171, Berlin, 2003.
7. D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, 2003.
8. D. Comaniciu, P. Meer, “Mean shift analysis and applications,” *IEEE Int. Conf. Computer Vision (ICCV'99)*, pp.1197-1203, Kerkyra, Greece, 1999.