2002 16th International Conference on Pattern Recognition

Vol. 2 (B)

Object Segmentation and Tracking Using Video Locales

James Au
UBVideo Inc.
Vancouver, BC, Canada
james@ubvideo.com

Ze-Nian Li
School of Computing Science
Simon Fraser University
Vancouver, BC, Canada
li@cs.sfu.ca

Mark S. Drew
School of Computing Science
Simon Fraser University
Vancouver, BC, Canada
mark@cs.sfu.ca

ABSTRACT

In this paper, we present a new technique based on feature localization for segmenting and tracking objects in videos. A *video locale* is a sequence of image feature locales that share similar features (color, texture, shape, and motion) in the spatio-temporal domain of videos. Image feature locales are grown from tiles (blocks of pixels) and can be non-disjoint and non-connected. To exploit the temporal redundancy in digital videos, two algorithms (intra-frame and inter-frame) are used to grow locales efficiently. Multiple motion tracking is achieved by tracking and performing tile-based dominant motion estimation for each locale separately.

1. INTRODUCTION

In many applications, the ability to automatically locate and track objects in videos is very important. The most intuitive way of accomplishing this task is to first generate temporally-tracked homogeneous regions and then apply further processing (automatic or human-aided) to identify the semantic objects.

In [8], we proposed *Feature Localization* as an alternative to traditional *image segmentation* in respect to object-based image retrieval. *Locales* are enclosures of local features that are not required to be connected, disjoint, or complete. Locales also operate at a higher level than pixels, as the basic building units used are *tiles* that correspond to blocks of pixels (e.g., 16×16 or 8×8). Localization is not merely a reduced-resolution method; pixel-based statistics are collected and used throughout the process without loss of high-resolution details. One of the advantages of localization is that it generates coarse locales that are more robust to noise and complex object surfaces. It is also much more attainable as it does not require complete pixel-level segmentation.

In this paper, we describe *video locales* that exploit the temporal redundancy in digital videos. The new method deals with *object segmentation and tracking* which are important issues for object-based video coding. Initially, we use a pyramidal probabilistic-unforced-linking algorithm to extract high quality color locales (using color as the main feature) within a video frame. For subsequent frames, a modified algorithm is used to exploit the similarities between consecutive video frames. Finally, extracted locales from consecutive frames are matched and dominant motion estimation is carried out for each matching pair of locales to provide temporal tracking. Since each locale is tracked separately, the method has the potential to tackle motions of multiple objects. Our motion estimation algorithm also follows the philosophy that all calculations are done based on tiles while keeping pixel-unit precision; hence we gain the advantage of utilizing both global and local information. Experiments have shown impressive results.

2. FEATURE LOCALIZATION CONCEPT

Feature localization was introduced and described in [8].

Definition 1: A *locale* L_f is a local enclosure of feature f.

A locale L_f uses blocks of pixels called *tiles* as its building units, and has the following descriptors:

1. Envelope $\Re_f$: a set of tiles representing the locality of L_f.
2. Geometry: mass $M(L_f)$, centroid $C(L_f)$, and pixel variances.
3. Color, texture, and shape parameters of the locale.

After a localization process the following is often true:

1. Locales are not connected: $\exists f$: L_f is not connected.
2. Locales are non-disjoint: $\exists f \exists g : L_f \cap L_g \neq \emptyset, f \neq g$.
3. Non-completeness: $\cup_f L_f \neq I$, not all pixels are represented.

3. VIDEO LOCALES ALGORITHM

Our approach is to create video locales that are coarse but accurate approximations for both object locations and movements. A *video locale* is a sequence of image feature locales that share similar features (color, texture, shape, and motion) in the spatio-temporal domain of videos. Using color as the main feature to localize, we extract image locales from each frame and then perform motion estimation based on tiles (while making use of pixel-precision statistics). We extend our localization algorithm to take advantage of temporal redundancies in videos.

There are 3 major components in the algorithm:

1. Intra-frame Locales generation using pyramidal probabilistic-unforced-linking.
2. Inter-frame Locales generation exploiting motion-compensated frame similarity for subsequent frames.
3. Locale motion estimation using tile-based energy minimization for 2D affine motion.

The Inter-frame algorithm automatically switches to Intra-frame processing if scene and shot changes occur (see Section 3.3).

3.1 Overall Algorithm

The overall algorithm for video object segmentation and tracking is as follows:

1. Read a frame from video.
2. If this is first frame,
   a. Generate **Intra-frame Locales**
3. Else
   a. Generate **Inter-frame Locales**
   b. **Match locales** from current frame to reference frame
   c. Perform **motion estimation** for each locale
4. Repeat 1 – 3 until end of video.

3.2 Intra-Frame Locale Generation

The intra-frame algorithm uses information within the video frame only. There are two major steps. Tiles are first generated

1051-4651/02 $17.00 © 2002 IEEE

from the image and then they are linked into locales in a pyramidal scheme.

3.2.1 Tile Generation

A relatively simple histogram analysis is performed to estimate the different dominant colors in a block of pixels. For each tile, an RGB histogram with bin widths 32 × 32 × 32 is constructed and the 5 most frequent color bins become the dominant colors for the tile. After the dominant colors are determined, each pixel in the tile is assigned to the closest dominant color. Each dominant color becomes a tile-feature and the average RGB is used as the final color. Geometrical statistics mentioned in Section 2 are calculated for each tile-feature. If there are more than one tile-feature in a tile, we say that the tile-features are overlapped.

Integer RGB values are used in this step for computational simplicity. Although RGB is not as perceptually accurate as other measures, we find that it gives good enough estimates. In subsequent processing, chromaticity and luminance are used instead.

Pseudo-code for the entire tile generation process and transitional/noise pixel determination is outlined in Procedure 1 and 2 respectively.

Procedure 1: Tile Creation

1. Calculate Dominant Colors:
   a. Create an RGB histogram (32 × 32 × 32 RGB bins) for the non-transitional and non-noise pixels.
   b. Merge color bins using intensity and chromaticity.
   c. 5 most frequent color bins are the *Dominant Colors*.
2. Assign each pixel (incl. transitional & noise) to closest dominant color.
3. A Tile-Feature (Locale) is created for each dominant color.

Procedure 2: Noise/Transitional Pixel Determination

1. Transitional: intensity is on a steep slope of any 2 of 8 neighbour pixels.
2. Noise: < than 2 neighbours with similar intensity.

3.2.2 Locale Growing Pyramid

In previous papers [8] we employed a bottom-up pyramid linking method for merging the tiles into locales; however, this is plagued by one of the most confounding problems for bottom-up image segmentation methods: forced-decisions with local or incomplete information. Most decisions have to be made at the lowest level, which is characterized by local and noisy data, and wrong decisions propagate through the pyramid to cause inaccurate results. In [5], T. H. Hong and A. Rosenfeld introduced an unforced-linking algorithm in which classification decisions are deferred until more information is available. Instead of forcing a pixel in the lower level to link with one parent in the upper level, the pixel is linked to multiple parents with probabilistic weights derived from their geometric and intensity similarities. Multiple iterations update the weights until they finalize. Pixels are then assigned to their most strongly linked parent (region). The advantage of this method is that errors made at low levels can be recovered after a few iterations.

This is the strategy we employ where overlapping tile-features and locales are used in the place of pixels. Parent locale statistics are updated using probabilistic weighting for two purposes: (1) reduce impact of outliers and noise, and (2) propagate weights to next level. When a child locale is merged into the parent locale wrongly or as an outlier, the link probability is low so it would not contribute as much to the parent locale as others. Note that each child locale can be compared to more than 4 parents because parent locales may overlap (each node is a list of locales); hence, as we move up in pyramid levels, tile-details (which are measured with pixel precision) are not lost while global information is gained. This is the main reason why this algorithm is not merely a multi-resolution analysis.

3.3 Inter-Frame Locale Generation

Discrepancies between two consecutive frames are mostly caused by four things: noise, object and camera movements, introduction of new objects, and special visual effects including cut, wipe and dissolve transitions. Since locales are fairly robust to small noise, we do not worry about noise. Object and camera movements account for most of the changes in videos. Our goal is to model and predict these movements. Motion estimation enables us to generate inter-frame locales quickly and at the same time provides locale tracking. The other two causes cannot be predicted so we always scan all pixels at least once to update such changes.

In the intra-frame algorithm, most of the computation time is spent on calculating dominant colors within the tiles and on iteratively linking and re-linking nodes at the bottom-most level in the pyramid. Dominant colors calculation is costly because it is mostly a pixel-level computation. Similarly, the bottom pyramid level has as many nodes as tiles.

Our strategy is to minimize those two steps. Assume that we have 2D affine motion estimation for each locale on the reference frame (discussion for motion estimation is deferred until Section 3.4). We first transform the envelopes of the locales onto the current frame according to their motion estimation. This prediction carries with it two major pieces of information for each tile in the current frame: predicted dominant colors and predicted tile-feature ownership. Each locale appends its color to the dominant color list of the tiles within its envelope, and all tile-features created for that color are assumed to link to that locale; hence, dominant color calculation is skipped, and the unforced-linking step is also skipped. If the predicted motion is 100% accurate, this will produce optimal results with almost no work.

After the dominant colors are predicted, we examine each pixel in each tile, update tile-features' statistics, and extract any unpredicted tile-features (new colours). These new tile-features are then linked into locales in a partial pyramid that involves only the new tile-features. The new locales are merged or appended to the predicted locales. Scene changes or large occlusions will generate many unpredicted tile-features, and full pyramid linking (as in Intra-processing) is performed in such cases.

Procedure 3 shows pseudo-code for the algorithm used for generating inter-frame locales. Inter-frame quality is almost identical to intra-frame but the average inter-frame computation time is only 25% of that of the intra-frame. For frames that have little motion or those that contain very predictable movements, computation times are as low as 10% of the intra-frame time.

Procedure 3: Inter-frame Locales Algorithm

1. Predict a locale for each reference locale in previous frame.
2. Update Tiles
   a. For each reference locale that is big enough:
      i. Predict envelope from motion vector in last frame.
      ii. Append its color to all intersecting tiles.

b. For each tile:
   i. Assign each pixel to closest predicted dominant color and keep top 5.
   ii. Each tile-feature is predicted to belong to the color's corresponding locale.
   iii. If more than 20% pixels have unpredicted colors, recalculate tiles as in Intra and recover ownership:
      1. For each tile-feature, match it to closest dominant color list and link it to corresponding locale.
      2. Count # tiles that contain unclaimed tile-features.
3. Locales Linking
   a. If # unpredicted tiles < ¼ of image
      i. Pyramid-grow the unclaimed tile-features.
      ii. Merge the grown locales with predicted locales.
   b. Else, re-grow all tiles as in intra-frame.

3.4 Locale Motion Estimation

The simplest way to estimating locale motion is locale-centroid displacement. The vector given by the movements of a locale's centroid provides a rough estimation to the translation vector for the whole locale; however, a two-dimensional translation model may not be sufficient in the presence of scaling and rotation. Furthermore, centroids are calculated as an average and therefore they are sensitive to outliers. It is more intuitive to examine the locale envelope directly as the goal is to predict how the envelope transforms across time.

Many motion segmentation algorithms exist in the literature. There are two main approaches: dominant motion method that derives motion directly from spatiotemporal image intensity gradient information; and indirect methods that first estimate optical flow field and then determine parameters and support for multiple motion models, such as motion parameter clustering, maximum likelihood (ML) segmentation, and maximum *a posteriori* probability (MAP) segmentation [11].

In this experiment, we adopt the dominant motion approach using over-lapping tiles as the base unit to extract a good approximation for the overall motion of each locale. It is possible that a single locale may exhibit multiple motions, especially locales from non-rigid and articulate objects. The dominant motion approach provides a very intuitive and direct way of extracting the overall movement.

3.4.1 Matching Locales

Since we are tracking color locales, we make the assumption that we can model most changes in the locale envelopes by a 2D affine model. This is usually the case when the 3D scene is sufficiently distant from the camera, which is quite common in most videos [7]. In this paper, we use the 6-parameter affine model.

Suppose we call the affine transform $a_x, b_x, c_x, a_y, b_y, c_y$, then the 2-D motion $(\Delta x, \Delta y)$ of a pixel is:

$$\Delta x = a_x x + b_x y + c_x, \Delta y = a_y x + b_y y + c_y \qquad (3.5)$$

If we assume image motion is small, then a Taylor series expanded only to linear terms is expected to be sufficiently accurate to describe the image motion. This results in the optical flow equations [6, 9]. For an affine motion, the least-square solution over a region of interest is the solution of a 6×6 matrix times the unknown 6-vector ($a_x, b_x, c_x, a_y, b_y, c_y$), with the right-hand-side being a forcing vector.

Since locale dominant motion estimation is carried out for each locale separately, only the envelope of one object is considered and the dominant motion assumption should hold valid for most cases. It is rare that an individual object will exhibit equal competing motions. Also, the iterative process and change detection can be avoided because there is only one motion to extract. Pixel-noise is minimized when tiles are used as the observation points. Finally, in the case of low spatial gradient regions, this can be detected by a motion reliability measure defined in [7] and we can recover by falling back on locale-centroid displacement to give us an estimation of the translation vector. Procedure 4 illustrates the pseudo-code for the motion estimation process.

Procedure 4: Locale Motion Estimation

1. For each (locale, reference locale) pair:
   a. If both Locale and reference Locale have > 5 tiles:
      i. Region of interest = union of tile envelopes
      ii. Estimate Dominant Translation (d_x, d_y)
      iii. Translate Reference locale tiles by (d_x, d_y)
      iv. Estimate Affine Transform $a_x, b_x, c_x, a_y, b_y, c_y$
      v. Compose the two transforms
      vi. If near-singular condition, go to step b.
   b. Else:
      i. (c_x, c_y) = Centroid displacement.

4. EXPERIMENTAL RESULTS

We have implemented the above algorithms in our system which can read any video formats and display locales in real-time. Figure 1 compares the results of inter-frame algorithm with those of intra-frame algorithm. The difference in quality is very small even though on average the Inter-frame algorithm is 70% faster than the intra-frame algorithm (see Table 1). On a modern PC our algorithm should be able to run at 20-30 frames per second.

We evaluate the quality of our motion estimation algorithm by monitoring the number of unpredicted tiles. The tile-based dominant motion estimation algorithm (DM) is compared against other methods: (1) assume all objects are stationary (SA), (2) predict translation by locale-centroid displacement (CD). Table 2 summarizes the results. As expected, stationary assumption method performs the worst while tile-based dominant motion estimation has the best performance. When motion cannot be well predicted by the affine model (as in the wiggling fish scene), centroid displacement is as good as dominant motion estimation.

In Figure 2, a video sequence with lots of motion and occlusions is tested under the full algorithm (intra-frame & inter-frame). Color localization is very effective even though it is a difficult video. Figure 3 shows the tracking of a hockey player by his blue jersey. Notice that the localization approach overcomes the problems of disconnected regions (stripes, logos, jersey) and complex motion to provide good tracking result.

5. CONCLUSION

In this paper, we presented a new algorithm for video object segmentation and tracking in a localization framework. An unforced-linking pyramidal scheme is used to achieve good localization, and temporal redundancy is exploited by an inter-frame algorithm. Fast, effective, object-based dominant motion estimation is possible because of the introduction of video locales. Experimental results show that video objects from a variety of natural scenes can be effectively segmented and tracked.

Potential applications of this work include object segmentation and tracking in sports coverage and surveillance, video summarization, and object-based video coding.

Many future improvements are possible. Currently the motion estimation does not address the merge/split problem. If a locale is split, only part of the envelope is used in estimating motion. In future we will investigate ways of allowing multiple-locales matching to track down all the various parts of a split or merged-locale. An adaptive threshold scheme will also help.

6. REFERENCES

[1] D.H. Ballard, C.M. Brown. *Computer Vision*. Prentice Hall, 1982.

[2] J. R. Bergen, E.H. Adelson, "Hierarchical, computationally efficient motion estimation algorithm," *J. Opt. Soc. Am.*., 4:35, 1997.

[3] P.J. Burt, C. Yen, and X.Xu, "Multi-resolution flow through motion analysis," *CVPR83*, pp. 246-252, 1983.

[4] M.S. Drew, J. Wei and Z.N. Li, "Illumination-invariant color object recognition via compressed chromaticity histograms of color-channel-normalized images", *ICCV98*, pp. 533-540, 1998.

[5] Hong, Rosenfield, "Compact Region Extraction Using Weighted Pixel Linking in a Pyramid", *IEEE PAMI*, 6(2), pp. 222-228, 1984.

[6] B.K.P. Horn, *Robot Vision*, MIT Press, 1986.

[7] M. Irani, B. Rousso, and S. Peleg, "Computing occluding and transparent motions," *Int. J. Comput. Vision* 12, pp. 5-16, 1994.

[8] Z.N. Li , O.R. Zaiane, Z. Tauber, "Illumination invariance and object model in content-based image and video retrieval", *J. Vis. Commun. Image Rep.*, Vol. 10, No. 3, pp. 219-244, 1999.

[9] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", *Im. Und. Wk.*, pp.121-130, 1981.

[10] P. Salembier and F. Marques. "Region-based representations of image and video: segmentation tools for multimedia services". *IEEE Trans. Circ. Sys. Vid. Tech.*, 9(8), pp. 1147-1169, 1999.

[11] M. Tekalp, "Chap. 4.9:Video Segmentation", *Handbook of Image and Video Processing*, Academic, pp. 383-399, 2000.

Table 1. Timing on Pentium II 400 (ms per frame)

~800 320x240 frames	Intra-frame	Inter-frame
Mean	330.3	95.9
Minimum	208.0	37.0
Maximum	587.0	455.0
Standard Deviation	95.5	42.0

Table 2. Motion Estimation Performance:

Scene Sequences	Method	Unpredicted Tiles per frame	% of Total Tiles (1600)
Simple linear motion (Scrolling texts)	SA	249.52	15.595%
	CD	0.34	0.022%
	DM	0.24	0.015%
Non-linear motion (Wiggling fish)	SA	177.89	11.118%
	CD	56.44	3.528%
	DM	57.51	3.594%
High motion & Complex scene (Football)	SA	342.75	21.422%
	CD	105.65	6.603%
	DM	83.66	5.229%

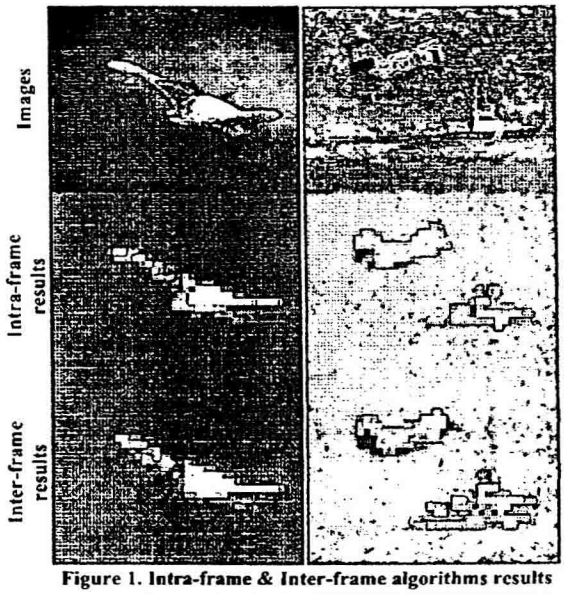

Figure 1. Intra-frame & Inter-frame algorithms results

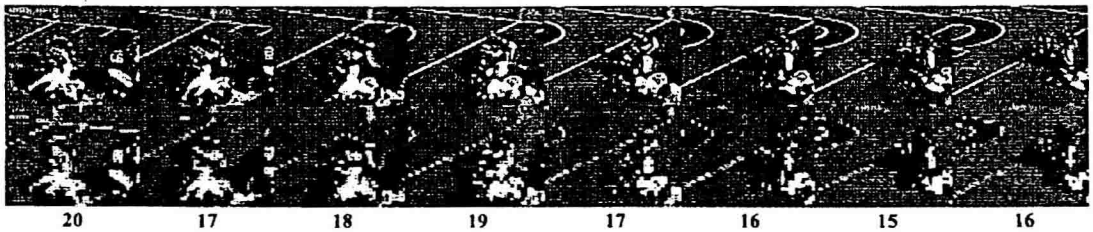

Figure 2. Video sequences: original frames are shown on top, followed by composed locales. The numbers below the sequences show the number of locales in each frame.

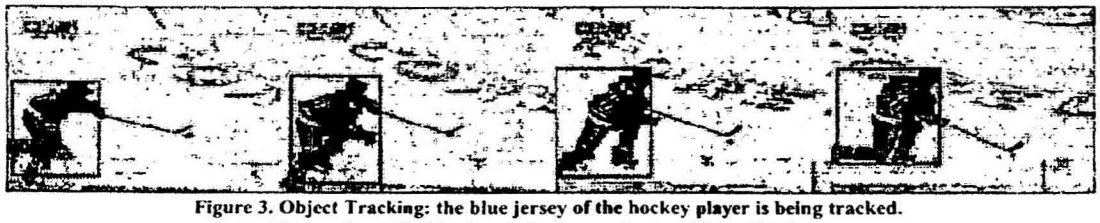

Figure 3. Object Tracking: the blue jersey of the hockey player is being tracked. Figure shows the blue locale by its envelope and bounding box.

Motion Texture: A New Motion Based Video Representation

Yu-Fei Ma, Hong-Jiang Zhang
Microsoft Research Asia
3/F Beijing Sigma Center, 49 Zhichun Road, Beijing, P.R.China (100080)
{yfma, hjzhang}@microsoft.com

Abstract

Motion is an important cue for video content perception. However, the lacking of effective motion representation becomes a barrier for automatic video content analysis. In this paper, we propose a new motion descriptor to capture motion pattern from video clip. First, we transform motion vector field to a number of directional slices of energy. Then, these slices are measured by a set of moments. As a result, a multi-dimensional vector, called Motion Texture, is formed. The effectiveness and efficiency of the proposed representation had been validated by motion-based shot retrieval experiments.

1. Introduction

To extract efficient motion representation from video clip still remains a challenging issue in the filed of video content analysis. It is not only because motion is a complex mixture of object and camera motions, but also because motion is a kind of information hidden in the temporal variances of other visual features, e.g., color, shape and texture. Without an effective representation, it is difficult to fully utilize motion information in automatic video content analysis, including content-based video retrieval, video classification, events detection, and so on.

Motion estimation is a conventional method to extract motion information from two consecutive frames [1]. Parametric global motion estimation generates the parametric model of camera motion or dominant motion, such as affine model. While non-parametric motion estimation generates a field of displacement pairs, such as optical flow. Both results of the two methods can be used as motion representations directly. For example, the optical flow is used for video indexing in [2]. However, the former is much coarse, and the latter is over fine for describing motion characters. Since human usually only concerns object motion, and cannot be aware of the existence of camera motion, one of semantic motion representations is the exact trajectory of moving object. In order to extract motion trajectories, the technologies of object segmentation and tracking [3], or motion layer extraction [4] are often adopted. However, the unreliability of automatic objects extraction makes it impossible to obtain exact object trajectories in the most of cases. So, as a simplified alternative, the motion regions' trajectories are extracted and applied to video retrieval in [5]. Another motion extraction method is based on the temporal slices of image volume [6, 7, 8]. In [8], Ngo characterized motion using multiple slices and tensor measurement. The temporal slices encode rich motion clues which are very useful for specific motion characterization, but they also have many confusable visual patterns that cannot be suppressed easily.

In this paper, we present a new motion representation based on non-parametric motion estimation. By measuring the energy distribution, we obtain a multi-dimensional vector, called *Motion Texture*. The motion texture can capture the motion patterns in video clip compactly and effectively, which have been validated by motion based shot retrieval experiments.

The rest of this paper is organized as follows. First, the conception of motion texture is introduced in Section 2. Then, a similarity measure of motion texture is defined in Section 3. In Section 4, we compare the performance of the proposed motion descriptor with a conventional one in a shot retrieval experiments. Finally, Section 5 concludes the paper.

2. Motion Texture

Non-parametric motion estimation can be carried out at the different scales, either pixel-based or block-based. The former generates dense flow, while the latter generates sparse displacement pairs, which all can be looked upon as a set of motion vectors. The motion vectors in a frame is usually called motion vector field (MVF). In this paper, we adopt the MVF in MPEG stream as approximate block-based motion estimation. The motion texture is extracted from MVFs by two steps. First, we transform MVF to a number of directional slices of energy. Then we obtain a multi-dimensional vector, namely *Motion Texture*, by moments measuring.

1051-4651/02 $17.00 © 2002 IEEE

In MVF, let (i,j) be the position of macro blocks in raster scan order, and $V_{i,j}.(\Delta x_{i,j}, \Delta y_{i,j})$ be the motion vector of macro block $MB_{i,j}$. We define the energy $En_{i,j}$ in macro block $MB_{i,j}$ as following:

$$En_{i,j} = \sqrt{\Delta x_{i,j}^2 + \Delta y_{i,j}^2} \quad (1)$$

Since the patterns in original MVF are not salient enough, we map the energy in MVF to a unit circle. As shown in Figure 1, we construct rectangular coordinates at the center of MVF, and polar coordinates at the center of unit circle. The width and height of MVF are $2w$ and $2h$ respectively. If let $x_{i,j}$ and $y_{i,j}$ denote the position of macro block $MB_{i,j}$ in rectangular coordinates, the process of mapping the energy in a MVF to a unit circle can be defined as:

$$g(\rho,\theta) = \sum_{x_{i,j}=-w}^{+w} \sum_{y_{i,j}=-h}^{+h} En_{i,j} \quad if \begin{cases} \rho = \bar{r}_{i,j} \\ \theta = \alpha_{i,j} \end{cases} \quad (2)$$

where $g(\rho,\theta)$ is the energy distribution function of unit circle, $\bar{r}_{i,j} = \sqrt{x_{i,j}^2 + y_{i,j}^2} \Big/ \sqrt{w^2 + h^2}$ is the normalized distance from macro block $MB_{i,j}$ to the center of MVF, and $\alpha_{i,j} \in [0, 2\pi]$ is the orientation of motion vector $V_{i,j}$. We call this mapping process *Circular Mapping*, and call the mapped unit circle *Energy Unit Circle* (EUC). In EUC, both object motion and camera motion present distinctive patterns.

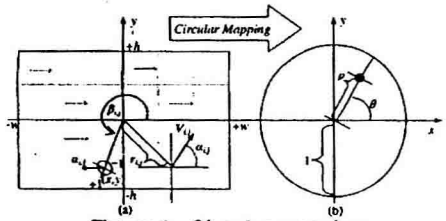

Figure 1. Circular mapping

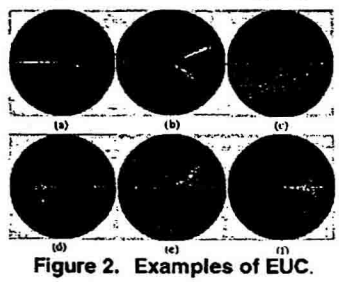

Figure 2. Examples of EUC.

Figure 2 gives some examples of EUC. (a) Camera panning right. There is a light line on the left. (b) Camera tracking. There are two light strips. The one extending to the brim of EUC results from the camera motion and the other with shorter length results from object motion. (c) Camera zooming. It presents a special pattern. (d) Irregular object motion with camera being static or moving slightly. (e) Object motion with specific pattern, moving along the orientation of $\pi/4$. (f) A special case, a cube turning around. Its pattern is very distinctive.

In order to capture the temporal pattern of motion during a period of time, we extract slices from the successive EUCs along temporal axis. As shown in Figure 3, we first divide EUC into n (n=4 in this paper) equiangular opposite sectors with the central lines at 0, $\pi/4$, $\pi/2$ and $3\pi/4$. Then the energy in each sector is accumulated to the central lines along homocentric circumference. Finally, we extract *directional slices* from EUC volume at those central lines. This process is called *directional slicing*.

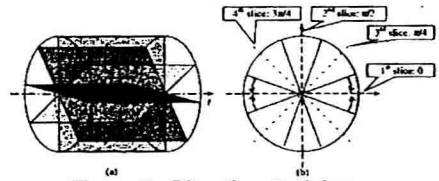

Figure 3. Directional slicing

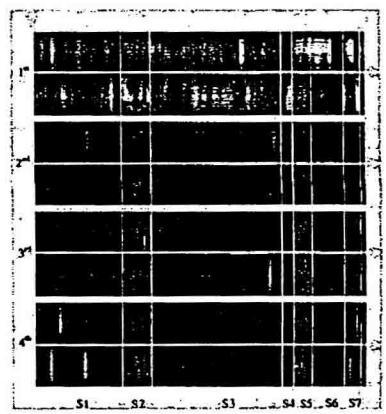

Figure 4. Directional slices samples from a segment of basketball game video

A segment of directional slices extracted from Basketball.mpg is shown in Figure 4, which has total 7 shots, S1~S7. The horizontal coordinate is temporal axis, and the vertical coordinate indicates the distance from macro block to the center of MVF. Shot-1 is a bout of offence with a shoot occurred (right court); In Shot-2, 5, and 7, camera is tracking a player. The energy evenly distributes among the 4 slices due to the irregular object motion; Shot-3 is another bout of offence with a shoot occurred (left court); Shot-4 is a specific wipe; and Shot-6 is also a bout of offence, but without shoot. In each slice, the positive and negative values of vertical coordinate denote the two opposite directions respectively, so 4 directional slices are able to describe the patterns in 8 directions. In

this way, motion intensity, dominant direction, and motion pattern all can be presented on a few gray-level images via the spatial and temporal distribution of energy.

The operations above reveal salient motion patterns from MVF. In order to obtain a compact and quantified representation, an effective measure of slice images is also required. According to Hu's *Uniqueness Theorem* [9], if a function $f(x,y)$ is piecewise continues and has non-zero values only in the finite region of the (x,y) plane, then the moments of all orders exist. It can be shown that the moment set $\{m_{pq}\}$ is uniquely determined by $f(x,y)$ and conversely, $f(x,y)$ is uniquely determined by $\{m_{pq}\}$. Therefore, if we describe the directional slices with energy density functions $f_n(x,y)$, the moments can be employed to measure these slices. Since the directional slice images $f_n(x,y)$ have finite area and, in the worst case, are piecewise continues, moments of all orders exist and a moment set will uniquely describe the information contained in them. In this paper, we select a sub set of moments from the zeroth to the fourth order to characterize slices. Assuming the slice images have the size of $M \times N$, the moments can be computed by (3)

$$m_{pq} = \sum_{x=0}^{N-1}\sum_{x=0}^{M-1} x^p y^q f(x,y) \tag{3}$$

where (p,q) = {(0,0), (1,0), (0,1), (2,0), (0,2), (3,0), (0,3), (4,0), (0,4)}. Based on these moments, we compute 9 values with specific physical meanings, including mass (m_{00}), the center of mass (4), the radii of gyrations (6), skewness (8) and kurtosis (9). Among them, the center of mass and the radii of gyrations need the normalization by the size of slices (5) (7).

$$COM_x = \frac{m_{10}}{m_{00}}, \quad COM_y = \frac{m_{01}}{m_{00}} \tag{4}$$

$$\overline{COM_x} = \frac{COM_x}{M}, \quad \overline{COM_y} = \frac{COM_y}{N} \tag{5}$$

$$ROG_x = \sqrt{\frac{m_{20}}{m_{00}}}, \quad ROG_y = \sqrt{\frac{m_{02}}{m_{00}}} \tag{6}$$

$$\overline{ROG_x} = \frac{ROG_x}{M}, \quad \overline{ROG_y} = \frac{ROG_y}{N} \tag{7}$$

$$Sk_x = \frac{\mu_{30}}{\mu_{20}^{3/2}}, \quad Sk_y = \frac{\mu_{03}}{\mu_{02}^{3/2}} \tag{8}$$

$$K_x = \frac{\mu_{40}}{\mu_{20}^2} - 3, \quad K_y = \frac{\mu_{04}}{\mu_{02}^2} - 3 \tag{9}$$

In (8)(9), the central moment μ_{pq} are computed by (10):

$$\mu_{pq} = \sum_{x=0}^{N-1}\sum_{x=0}^{M-1} (x - COM_x)^p (y - COM_y)^q f(x,y) \tag{10}$$

where (p,q) = {(2,0), (0,2), (3,0), (0,3), (4,0), (0,4)}. In order to characterize the motion's convergence or divergence relative to FOE, a *signed energy* is defined for each macro block $MB_{i,j}$:

$$SEn_{i,j} = \begin{cases} 1 & (|\alpha_{i,j} - \beta_{i,j}| < \frac{\pi}{2}) \\ -1 & otherwise \end{cases} \tag{11}$$

where $\alpha_{i,j}$ still denotes the orientation of motion vector $V_{i,j}$, and $\beta_{i,j}$ is the direction angle of the macro block $MB_{i,j}$ in MVF, as shown in Figure 1 (a). The signed energy is also transformed to the directional slices by circular mapping and directional slicing. Then we compute the average signed energy in each directional slice by (12).

$$\overline{SEn} = \frac{1}{M \times N}\sum_{x=0}^{N-1}\sum_{x=0}^{M-1} SEn, \tag{12}$$

With nine energy distribution measures and one additional signed energy measure of each directional slice, a $10 \times D$ dimensions vector: $T=\{T^0, T^1, T^n, ..., T^{D-1}\}$ is obtained, where $T^n = \{m_{00}^n, \overline{COM_x}^n, \overline{COM_y}^n, \overline{ROG_x}^n, \overline{ROG_y}^n, Sk_x^n, Sk_y^n, K_x^n, K_y^n, \overline{SEn}^n\}$ and $n \in [0, D-1]$. We name this vector *Motion Texture*, by which all of motion characteristics in video clip can be represented compactly and effectively. Since $D=4$ in this paper, we obtain a 40-dimentional vector.

3. Similarity Measurement

To apply a descriptor in matching, the similarity also should be formulated. In this section, we define a similarity measure for motion texture. Since the dynamic range of each component of motion texture is quit different, the normalization is indispensable when we compare two motion texture vectors. Assuming we have a video clip database, the motion texture is extracted from each clip. Then we normalize each component of vectors by the inverse of the standard variance. The standard variance of k^{th} component is

$$\sigma_k = \sqrt{\frac{\sum_{l=1}^{L}\left(v_k^{(l)} - \bar{v}_k\right)^2}{L}} \tag{13}$$

where $v_k^{(l)}$ denotes the k^{th} component of the l^{th} feature vector, L is the number of samples in database, and $\bar{v}_k$ is the mean of $v_k^{(l)}$, which is computed as

$$\bar{v}_k = \frac{\sum_{l=1}^{L} v_i^{(l)}}{L} \tag{14}$$

By using the normalization coefficients as weights, we adopt weighted Euclidean distance to measure the similarity of motion texture. When comparing two motion texture vectors T^a and T^b in video clip database, the similarity is defined as following:

$$Sim\left(T^a, T^b\right) = \sqrt{\sum_{k=1}^{10\times D}\frac{1}{\sigma_k^2}\left(v_k^a - v_k^b\right)^2} \tag{15}$$

In this way, the modality-dependent amplitude difference is reduced effectively. The effectiveness of this similarity measurement has been verified by our experiments.

4. Experiments

We evaluated the proposed representation and similarity measure on a motion-based shot retrieval system. A test video database were built with about 10 hours' real-world videos, including documentary, sports and news. These videos were segmented into shots by an automatic algorithm. Consequently, there are 10000 shots in the test video database. For comparison purpose, we also implement a conventional method based on motion intensity and dominate directions. From each shot, the motion features used in the two methods are extracted. According to the similar motion patterns, we manually classify the shots in video database into 54 classes offline. First, we discriminate shots with some patterns from the ones without any salient motion pattern. The latter is quantified into 5 levels, 0–4, based on motion intensity. On the other hand, the shots with patterns are classified into 16 classes of camera pattern and 43 classes of object pattern. The number of camera pattern classes is fixed, such as panning left/right, zooming-in/out, etc. The number of object pattern classes is variable, since the motion patterns of the objects in real world can not be totally enumerated. For example, diving, high jumping, shooting, etc. are all have their own motion pattern. Then 14 classes with enough elements (>200 shots) are selected as our test set, including 4 classes of non-pattern (NP), 8 classes of object pattern (OP), and 2 classes of camera pattern (CP). In each class, one of members is picked out as query sample in turn, and the rest members are used as ground truth. The performance is evaluated by *average normalized modified retrieval rank* (***ANMRR***) and *average retrieval rank* (***ARR***), which were proposed in [10]. The experimental results are listed in Table 1. The lower the *ANMRR* value is, the better the performance of retrieval is. Contrarily, the higher the ***ARR*** value, the better the performance.

Query	Conventional Method		Motion Texture based Method	
	ANMRR	ARR	ANMRR	ARR
NP-1	0.6765	0.4998	0.5333	0.6065
NP-2	0.6603	0.4982	0.5011	0.6754
NP-3	0.6499	0.4371	0.5027	0.6508
NP-4	0.6598	0.4009	0.5249	0.6192
OP-1	0.4867	0.5865	0.2325	0.8154
OP-2	0.3944	0.7011	0.1305	0.8745
OP-3	0.3293	0.6581	0.0992	0.9046
OP-4	0.3943	0.6402	0.1375	0.8959
OP-5	0.4149	0.6749	0.2043	0.8076
OP-6	0.5061	0.6574	0.1214	0.8832
OP-7	0.2012	0.7625	0.0000	1.0000
OP-8	0.2835	0.8081	0.0000	1.0000
CP-1	0.1628	0.7963	0.0825	1.0000
CP-2	0.2037	0.8755	0.0933	1.0000
Avg.	**0.4302**	**0.6426**	**0.2259**	**0.8380**

Table 1. Performance Evaluation

Table 1 shows that the motion texture based method always outperforms the conventional method, especially for the shots with salient object motion patterns. Since the camera motions are very distinctive, they can be identified by both methods easily. In the case of shots without salient patterns, namely non-pattern, the improvement by the proposed method is limited, because only motion intensity and dominant direction are contributing. When motion texture is extracted from the MVF in MPGE stream, the speed of extraction can be much faster than real-time.

5. Conclusions

In this paper, we have presented a generic motion representation, *Motion Texture*. Most of motion characteristics, such as motion intensity, dominant directions, spatial and temporal motion patterns, can be fully characterized in this representation. The experimental results indicate that *Motion Texture* is very effective and efficient for motion pattern based shot retrieval. Also, *Motion Texture* can be used in other motion related applications, such as video classification, event detection, and surveillance, etc., which are our future work.

References

[1] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performances of optical flow techniques," Int. Journal of Computer Vision, 12:34, 1994.

[2] E. Ardizzone, M.L. Cascia, "Video Indexing Using Optical Flow Field," *proceeding of ICIP'96*, pp. 831-834, 1996.

[3] D. Zhong, S.F. Chang, "AMOS: an active system for MPEG-4 video object segmentation," *Proc. Image Processing*, 1998.

[4] H.S. Sawhney, S. Ayer, "Compact Representations of Videos through Dominant and Multiple Motion Estimation," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp.814-830, 1998.

[5] S. F. Chang, *et al*, "VideoQ: An Automated Content Based Video Search System Using Visual Cues", *Proceeding of ACM Multimedia*, pp.313-324, 1997

[6] S.L. Peng, "Temporal Slice Analysis of Image Sequences," *proceeding of Computer Vision and Pattern Recognition*, pp.283-288, 1991.

[7] F. Liu, R.W. Picard, "Finding Periodicity in Space and Time," *Proc. IEEE intl. Conf. On Computer Vision*, pp. 376-383, 1998.

[8] C.W. Ngo, *et.al.*, "Motion Characterization by Temporal Slices Analysis," *proceeding of Computer Vision and Pattern Recognition*, pp.768-773, 2000.

[9] M.K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Tans. on Information Theory*, vol. IT-8, pp.179-187, February 1962.

[10] "MPEG-7 Visual part of eXperimentation Model (XM) Version 2.0," *MPEG-7 Output Document ISO/MPEG*, Dec. 1999.

Real-Time MPEG2 Video Watermarking in the VLC Domain

Chun-Shien Lu†, Jan-Ru Chen‡, Hong-Yuan Mark Liao†, and Kuo-Chih Fan‡

†Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC
‡Dept. Comput. Sci. and Infor. Eng., National Central Uni., Taiwan, ROC
Email: {lcs, philips, liao}@iis.sinica.edu.tw; kcfan@csie.ncu.edu.tw

Abstract

This paper proposes a compressed domain video watermarking scheme for copyright protection. Our scheme is designed based on the concept of communications with side information. For making the real-time detection a reality, the watermark is directly embedded and detected in the VLC domain. The typical problems of video watermarking such as preservation of bit rate, video attacks, real-time detection will be examined. The performance of the new watermarking scheme will be examined by checking its robustness capability against attacks together with false positive analysis.

1 Introduction

Watermarking has received much attention due to the popularity of data communication through the Internet. Among various media data, digital video is the one that carries the most amount of data. Hence, it is not easy and realistic to embed/detect watermarks directly in a raw video [6] in real time. Usually, a raw video has to be compressed first and then watermarked before it is transmitted through the network. However, the major concern is how to design a feasible compressed video watermarking scheme such that the hidden watermarks could be detected in real time.

In the literature, only a few compressed video watermarking schemes were proposed [2, 3, 5]. In [2], the header/side information and motion vectors of an MPEG2 bitstream are not changed during watermarking. They arranged a watermark sequence to be two-dimensional and has the same size with a video frame. Then, the watermark signal is 8×8 DCT transformed and added into the DCT coefficients of a video stream. In other words, their compressed domain video watermarking is, in fact, performed in the DCT domain. Therefore, some preprocessing operations such as inverse entropy coding and inverse quantization are required. Besides, no attacks were tested in their experiments. In [3], Langelaar *et al.* proposed a video watermarking scheme performed in the compressed domain based on VLC codewords. At first, they divided run-level pairs into many groups with the same VLC codeword length under the constraint that the level difference in each group should be exactly one. During watermark embedding, a run-level pair was either unchanged or replaced depending on the incoming watermark value. Their method was basically a least significant bit (LSB)-type method. Recently, Langelaar *et al.* proposed a differential energy watermarking (DEW) algorithm [3] performed in the DCT domain. DEW means that watermark bits are inserted by removing the high-frequency DCT coefficients. The authors claimed that it is not possible to remove the DEW watermark without causing perceptual degradation.

From the above review, we know that a robust real-time video watermarking scheme which can be operated in the compressed domain is really necessary. In this work, we shall propose a solution to achieve the above goal. In order to make the proposed scheme more accurate and efficient, we shall put some major issues into consideration. First, the concept of viewing watermarking as communication with side information [1] will be adopted. Based on this concept, a new compressed video watermarking scheme is proposed. Our scheme will fully utilize the prior knowledge of a cover data such that the requirement of blind detection can be achieved by estimating the cover data from a suspect data.

2 The Communications with Side Information-based Watermarking Scheme

In this paper, the proposed video watermarking scheme is extended from [4], which was proposed for block-based spatial-domain image watermarking. Since a video is composed of a large number of frames and each frame is in fact an image, we shall start the discussion by talking about how to embed watermarks in an image.

2.1 Embedding Watermarks in Images

Let I be a cover image (or a frame of a video) with pixel values $x(j)$ $(1 \leq j \leq N \times M)$, where $N \times M$ is the image's

1051-4651/02 $17.00 © 2002 IEEE

size. The cover image is mean filtered to generate a new set of pixel values $\bar{x}(j)$. Let the difference between an original pixel $x(i)$ and its corresponding mean filtered pixel $\bar{x}(i)$ in a block b be

$$d(i) = x(i) - \bar{x}(i). \quad (1)$$

In the following, i indicates the pixel's index in a block. These $d(i)$ values will be normalized into a Gaussian distribution with zero mean and unit variance, i.e.,

$$d_G(i) = \frac{d(i) - \mu}{\sigma}, \quad (2)$$

where μ and σ are, respectively, the mean and the standard deviation of $d(i)$. To conceal a watermark like a communication process carrying side information [1], we replace $d_G(i)$ by $w(i)$ and get the modified $d(i)$ as

$$d^h(i) = w(i) \cdot \sigma + \mu. \quad (3)$$

Hence, a pixel value $x(i)$ is modulated as

$$x^h(i) = \bar{x}(i) + d^h(i) = \bar{x}(i) + w(i) \cdot \sigma + \mu. \quad (4)$$

In order to preserve certain fidelity, the modulation operation might be modified as follows:

$$x^h(i) = \bar{x}(i) + (w(i) \cdot \lambda + d_G(i) \cdot (1 - \lambda)) \cdot \sigma + \mu, \quad (5)$$

where λ is a weight used to adjust the compromise between robustness and fidelity.

2.2 Detecting Watermarks from Images

In the detection process, suppose $\lambda = 1$ and let a watermarked (but not attacked) image be first mean filtered. The mean filtered pixel value can be approximated by

$$\bar{x}^h(i) = \frac{1}{b_1 \times b_1} \sum_{t=1}^{b_1 \times b_1} (\bar{x}(t) + w(t) \cdot \sigma + \mu) \approx \bar{x}(i), \quad (6)$$

based on the characteristic of mean filtering (with window size $b_1 \times b_1$). From Eq. (6), the extracted signal s^e can be derived as follows:

$$s^e(i) = x^h(i) - \bar{x}^h(i) \approx w(i) \cdot \sigma + \mu. \quad (7)$$

The extracted signal $s^e(i)$ is equal to the $d^h(i)$ value in Eq. (3). This implies that the hidden watermark w could be perfectly recovered and explains why mean filtering is used.

3 Compressed Domain Video Watermarking

3.1 Video Watermark Embedding Process

3.1.1 Selection of Suitable Data

In the video decoding stage, a video bitstream is usually decoded into codewords by VLC decoding. Every codeword corresponds to a run-level pair denoted as (r, l). For a given run-level pair (r, l), the value of r indicates the *positions* of a set of DCT coefficients in a zigzag-scan order. In addition, the value of l represents the *magnitude* of a DCT coefficient only. More specifically, run (r) represents the number of DCT coefficients with magnitude zero preceding the current run-level pair and level (l) corresponds to the quantization value of the current DCT coefficient. If the run value (r) of a codeword has been changed, then the frequency in a zigzag-scan order is also significantly changed. In other words, the image would be distorted significantly because a great number of DCT coefficients have been forced to change. On the other hand, if a level value (l) is changed, then only one DCT coefficient will be changed. It will not affect other frequency components. Therefore, we understand that it is much easier to preserve the fidelity of a video by modifying the level values (l). Based on the above reasons, we have chosen to embed watermarks by modulating level values (l) instead of run values (r).

3.1.2 Selection of Suitable Position

We consider what kind of GOP structure in a compressed bitstream is suitable for watermarking. Basically, a GOP structure is composed of three kinds of frames: "I", "P", and "B." Owing to the data amount of the "B" and "P" frames are relatively fewer than that of the "I" frame and the reconstruction of "B" and "P" frames depends on the "I" frame, it is enough to embed watermarks on "I" frames only since the other non-"I" frames will inherit watermarks after decoding.

Since the size of a video bitstream is subject to change at different compression ratios, the number of run-level pairs will also be changed. This will cause the asynchronization problem in watermark detection. In order to tackle this problem, we propose to conceal a watermark value into a macroblock (MB) because the number of macroblocks is *invariant* to different attacks including video compressions. In addition, one watermark is inserted into the Y component of an I-frame so that the problem due to different sampling $(Y : Cb : Cr)$ can be avoided. In this paper, the length of a hidden watermark is equal to the number of macroblocks. This implies that one watermark bit will be embedded into a macroblock.

3.1.3 Embedding of a Watermark

Suppose there are in total N macroblocks in a video bitstream. Let (r_{ij}, l_{ij}) be the j-th run-level pair in the i-th macroblock, $u(i)$ be the mean of the levels, and n_i be the number of levels in the i-th macroblock. Under these circumstances, the mean values $u(i)$ $(1 \le i \le N)$ will form a 1-D sequence. Let $\bar{u}(i)$ be the mean filtered value obtained from $u(i)$. According to Eq. (1), we can obtain

$d(i) = u(i) - \bar{u}(i)$ with μ and σ being the mean and the standard deviation, respectively. Next, these $d(i)$ values can be normalized into $d_G(i)$ as in Eq. (2). Then, a watermark sequence $\mathbf{w} = \{w(1), w(2), ..., w(N)\}$ to be hidden is generated by a secret key. Every watermark value $w(i)$ $(1 \leq i \leq N)$ is embedded into a macroblock by replacing its corresponding $d_G(i)$ value, as Eq. (3) indicates.

So far, watermarking is actually not finished because only the quantity $d^h(i) - d(i)$, used to modulate the mean of level values, is obtained in a macroblock. We still need to modulate every run-level pair in a macroblock. In this paper, we propose to propagate the modulation quantity $d^h(i) - d(i)$ to all levels of the run-level pairs of a macroblock. That is, the original run-level pair (r_{ij}, l_{ij}) is modulated as (r_{ij}, l^h_{ij}) for $1 \leq j \leq n_i$ and $1 \leq i \leq N$, where $l^h_{ij} = l_{ij} + (d^h(i) - d(i))$. The reason behind this is that based on the use of mean filtering, it is easy to preserve the modulation quantity of each level value to be the same as that of their mean. This result facilitates the use of mean filtering for watermark embedding and detection. This implies that we have changed from macroblock-based modulation to level-based modulation and the watermarking operation is now actually accomplished.

In the process of compressed video watermarking, modulations of level values, in fact, correspond to modifications of quantization values obtained by applying a quantization table to the DCT coefficients. In order to maintain the fidelity of a watermarked video, it is necessary to modulate the corresponding DCT coefficient by at most one quantization value. On the other hand, in order to achieve the purpose of watermarking it requires to modulate the corresponding DCT coefficient by at least one quantization value. Therefore, the modulation quantity $|d^h(i) - d(i)|$ should satisfy the constraint:

$$|d^h(i) - d(i)| = 1 = |(w(i) - d_G(i)) \cdot \sigma + \mu|, \quad (8)$$

where 1 means one unit of a level value or a quantization value. From this constraint, we can further derive the embedded watermark value, $w^e(i)$, as

$$w^e(i) = \begin{cases} d_G(i) + \frac{1-\mu}{\sigma}, & \text{if } w(i) \geq 0, \\ d_G(i) - \frac{1+\mu}{\sigma}, & \text{if } w(i) < 0, \end{cases} \quad (9)$$

in order to guarantee the same sign between $w(i)$ and $w^e(i)$. The above mentioned modification is necessary because if $d_G(i)$ is completely replaced by $w(i)$, then DCT coefficients will be significantly modified and thus causes sensible visual defects. As a consequence, the watermark modulation strategy specified in Eq. (3) is used under the constraint that Eq. (9) holds (no longer depends on λ in Eq. (5)). This constraint guarantees the fidelity of a video. Under the above paradigm, the modified $d(i)$ value is obtained by

$$d^h(i) = w^e(i) \cdot \sigma + \mu, \quad (10)$$

which is slightly different from Eq. (3). Accordingly, the $u(i)$ value will be modulated as

$$u^h(i) = \bar{u}(i) + d^h(i). \quad (11)$$

It is noted that the insertion of $w^e(i)$ instead of $w(i)$ will play a trade-off role between robustness and transparency.

During the video watermarking in the VLC domain, we have found some problems that should be particularly addressed: (i) If (r_{ij}, l^h_{ij}) does not exist in the VLC codewords, then it will create the video coding and decoding problem. In order to avoid this problem, the modulated (r_{ij}, l^h_{ij}) should be enforced to stay with the value of its closest run-level pair in the VLC codewords. (ii) If the difference between the original level value and its corresponding modulated level value is large enough, the fidelity of a video would be hard to be preserved. Therefore, we have chosen to maintain the original run-level pair unchanged. On the other hand, if the modulated level l^h_{ij} is less than or equal to zero, then it is set to be 1 in order to maintain the correctness of decoding.

3.1.4 Problems in the Compressed Domain Watermarking

When watermarking is directly performed in the compressed video domain, some problems may occur. First, we should deal with the increase of bit-rate (BR) problem. In a compressed video sequence, the size of a bitstream is not allowed to increase after watermarking. Therefore, a trick is proposed to deal with this problem. We first check the total amount of size reduction by embedding negative watermark bits. Then, the total amount of size increased by embedding positive watermark bits must be controlled to be smaller than the amount of size reduction.

Secondly, our method embeds watermarks without considering the information of motion vectors because modulating motion vectors will incur significant perceptual modifications. However, even motion vectors are unchanged, the quality of a video will still be affected. This is because the subsequent frames are reconstructed based on referring a previously watermarked I-frame. In order to deal with this problem, we have extensively conducted a set of experiments and have found that all transparency degradations were occurred at the I-frames. However, when a video sequence is normally played, these degradations won't be sensed easily. Therefore, we won't spend extra energy to take care of this problem because it may increase the complexity of our designed algorithm.

3.2 Video Watermark Detection Process

3.2.1 Detection of the Watermark

From a suspect video, we first calculate its mean level values and mean filtered level values of macroblocks as $u^a(i)$ and

$\bar{u}^a(i)$ (for $1 \leq i \leq N$), respectively. In fact, the values, $\bar{u}^a(i)$, are the estimated cover video. Then, the extracted watermark values could be determined as

$$w^e(i) = sign(u^a(i) - \bar{u}^a(i)), \qquad (12)$$

which is basically an inverse operation of Eq. (11). The output of $sign$ function is $+1/-1$ if its argument is positive/negative. Finally, a normalized correlation value, derived by checking the relation between the original watermark $w(i)$ and the extracted watermark $w^e(i)$, is used to indicate the presence/absence of a hidden watermark if it is larger/smaller than a threshold.

4 Experimental Results

In our experiments, the video sequence, Flower-Garden, compressed at 15 Mbit/s was used. The frame size was 704×576. In average, the PSNR of the watermarked video was higher than 36dB. The original video and the watermarked video were visually indistinguishable. Some common attacks, including MPEG compression with bit rates of $6M$ bps, $4M$ and $2M$ bps, additive noise adding (27.05dB), sharpening, frame averaging, frame rate changing, mean filtering, and I-frame dropping+compression, were used for robustness test. Based on the obtained results, the threshold was determined as 0.137(0.150) if the desired false positive probability was $10^{-6}(10^{-7})$. Fig. 1 shows the correlation value with respect to each I-frame in some attacked videos and non-watermarked videos. It is noted that the correlation values detected from those watermarked/attacked video were easily separated from those detected from non-watermarked videos. This implies that both the false negative and the false positive probabilities derived by applying our method are low. Besides, special video attacks including collusion attacks and copy attacks have also been tested. Results have shown that our method is robust against collusion attacks but fragile to copy attacks if the added watermark is strong. As for time complexity, video decoding process spent 182 seconds but the watermark detection used 12 seconds, which implies that the detection time is negligible.

5 Conclusion

In this paper, we have demonstrated how robust video watermarking could be done in the VLC domain. Our method has been verified by many video attacks together with false positive analysis. From the current results, we have found that the correlation value detected from a watermarked (but not attacked) video is not extremely high. This is because there are about $20\% \sim 30\%$ macroblocks having near zero level values (corresponding to DCT coefficients after inverse quantization). If the embedding process is conducted on some small DCT coefficients, then the watermarked video won't be robust. Therefore, our future work will focus on improving the robustness by embedding watermarks on some strictly selected run-level pairs.

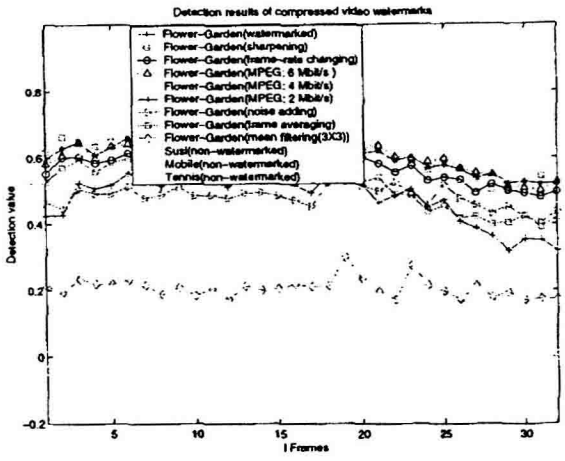

Figure 1. Watermark detection results.

References

[1] I. J. Cox, M. L. Miller, and A. McKellips, "Watermarking as Communications with Side Information", *Proc. of the IEEE*, Vol. 87, No. 7, pp. 1127-1141, 1999.

[2] F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," *Signal Processing*, Vol. 66, No. 3, pp. 283-302, 1998.

[3] G. C. Langelaar, I. Setyawan, and R. L. Lagendijk, "Watermarking Digital Image and Video Data," *IEEE Signal Processing Magzine*, Vol. 17, No. 5, pp. 20-46, 2000.

[4] C. S. Lu and H. Y. Mark Liao, "An Oblivious and Robust Watermarking Scheme Using Communications-with-Side-Information Mechanism", *Proc. IEEE Int. Conf. on Information Technology: Coding and Computing*, Las Vegas, Nevada, pp. 103-106, 2001.

[5] C. S. Lu, J. R. Chen, H. Y. Mark Liao, and K. C. Fan, "Watermarking on Compressed/Uncompressed Video Using Communications with Side Information Mechanism," in Timothy K. Shih (Eds.), Distributed Multimedia Databases: Techniques and Applications, Chap. 11, pp. 173-189, Idea Group Publishing, 2002.

[6] M. D. Swanson, B. Zhu, and A. H. Tewfik, "Multiresolution Scene-based Video Watermarking Using Perceptual Models." *IEEE Journal on Selected Area in Communications*, Vol. 16, No. 4, pp. 540-550, 1998.

A Novel Approach for Single View Based Plane Metrology

Guanghui Wang, Yihong Wu, Zhanyi Hu

National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences, Beijing 100080, P. R. China
{ghwang, yhwu, huzy}@nlpr.ia.ac.cn

Abstract

In this paper, a new approach is proposed for single view based plane metrology. The approach is based on a pair of vanishing points from two orthogonal sets of space parallel lines. Extensive experiments on simulated data as well as on real images showed that our new approach can achieve as good result as that of the homography based one which is widely used in the literature, but our new approach do not need any explicit specifications of space control points. Since in many real applications, particularly, in indoor environment, orthogonal lines are not rare, for example, a frame of window or a door, our new approach is of widely applicable.

1. Introduction

One of the main aims of Computer Vision is to take measurements of the environment and reconstruct its 3D model. The problem of using vision to measure world distance has attracted a lot of attention and received wide applications in recent years[1], such as architectural and indoor measurement, reconstruction from paintings, forensic measurement and traffic accident investigation. Traditional approach to measuring is to take all the distances manually by using metric tapes or rulers. This approach is time consuming, prone to errors and invasive. By computer vision based method, what one needs to do is only to take several pictures, then all measurements can be taken offline with more accuracy, flexibility and efficiency. In recent years, many researchers have been working on the problem and proposed some useful methods and algorithms[1~6].

Generally speaking, the methods in the literature may be divided into two categories. The classical one is to reconstruct the metric structure of the scene from 2 or more images taken from different point of view using stereo vision technique[1,5~6]. This is a hard task because it fundamentally involves solving the correspondence and camera calibration problems, and also, the method may subject to a loss of accuracy due to error propagation along the computation chain.

The other one is to directly use one uncalibrated image[1~3]. Only one view does not provide enough information for a complete 3D reconstruction. However some metrical quantities can be retrieved from the knowledge of some geometrical information such as the relative position of points, lines and planes in the scene. The methods in this category have been proved that they are easy to use and with equal accuracy. In[1,2], a homography based approach to calculate the Euclidean distance of two points on a world plane, as well as the uncertainty analysis of measurement was proposed. By using this method, at least the coordinates of four control points on the world plane and their corresponding image points should be known beforehand. It is obvious that the accuracy of this method largely depends on that of the control points measurements. But in some cases, it is really hard to obtain an accurate measurement of the key points on world plane. In[1], the authors described another approach to compute 3D affine measurement from a single perspective image. It is assumed that the vanishing line of a reference plane in the scene as well as a vanishing point in a reference direction (not parallel to the plane) maybe determined from the image, then three canonical type of measurements can be computed.

In this paper, we mainly focus on single view based plane metrology and propose a novel approach named vanishing points based method. In addition, a comparative study of the new methods, together with the homography based one, is carried out. The paper is organized as follows. In section 2, some preliminaries on homography and vanishing point are briefed. Then the novel method is elaborated in section 3. In section 4 and 5, the two methods are compared experimentally both on simulated data and real image. Some conclusions are given at the end of this paper.

2. Some preliminaries

2.1. Plane to plane homography

From perspective projection, a 3D point $\tilde{\mathbf{x}}$ in space is projected to an image point $\tilde{\mathbf{m}}$ via a 3×4 projection matrix $\mathbf{P}$ as

1051-4651/02 $17.00 © 2002 IEEE

$$s\tilde{\mathbf{m}} = \mathbf{P}\tilde{\mathbf{x}} = [\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3,\mathbf{p}_4]\tilde{\mathbf{x}} \quad (1)$$

where, $\tilde{\mathbf{m}}$ and $\tilde{\mathbf{x}}$ are homogeneous coordinates in the form of $\tilde{\mathbf{m}} = (u,v,w)^T$, $\tilde{\mathbf{x}} = (X,Y,Z,W)^T$, and s is a nonzero scalar. For 3D coplanar points, without loss of generality, we assume $Z = 0$, then

$$s\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{P}\begin{bmatrix} X \\ Y \\ 0 \\ W \end{bmatrix} = \underbrace{[\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_4]}_{\mathbf{H}}\begin{bmatrix} X \\ Y \\ W \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} X \\ Y \\ W \end{bmatrix} \quad (2)$$

Therefore, a 3D coplanar point and its corresponding image point can be related by the so-called plane to plane homography $\mathbf{H} = [\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_4]$. A homography $\mathbf{H}$ is described by a 3×3 non-singular matrix, with 8 degrees of freedom, because it can only be defined meaningfully up to a scale factor. According to equation (2), each image to world point correspondence can give two linear constraints to the 9 elements of homography, thus, given N (N=4) space coplanar points and their correspondences in image, the homography matrix can be uniquely calculated. If N>4, the matrix is over determined. For non-perfect data, $\mathbf{H}$ can be estimated by a suitable minimization scheme[1,7]. Finally, if we know the homography between the world and image plane, then an image point can be back-mapped into world point via $\mathbf{H}^{-1}$.

Depending on the conditions arising from different situations, the homography can also be estimated in different ways so as to increase its accuracy. For example, if we know a space line L and its corresponding image l under projective projection, then we have $s\mathbf{l} = \mathbf{H}^{-T}\mathbf{L}$, which can also provide two constraints on homography $\mathbf{H}$. So given at least 4 correspondences from coplanar space lines to image lines, the homography can also be computed.

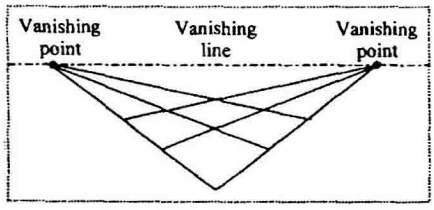

Fig.1 Vanishing point and Vanishing line

2.2. Vanishing point and vanishing line

Under perspective projection, parallel lines in 3D space project to converging lines in the image plane whose intersection, possibly located at infinity, is called vanishing point. Different sets of parallel lines on the same space plane define different vanishing points, all these vanishing points lie on the same line named vanishing line as shown in Fig.1. Vanishing points and vanishing lines convey a lot of information about direction of lines and orientation of planes in space, these entities can be estimated directly from images and no explicit knowledge of the relative geometry between camera and viewed scene is required. There are many methods for detection and calculation of vanishing points and vanishing lines in the literature[8,9]. If more than two lines are available, then a Maximum Likelihood Estimation algorithm (MLE) or least squares technique can be employed to estimate the vanishing point[1].

3. Vanishing points based distance measuring

In this section, we will describe an algebraic approach for distance measuring based on vanishing points. If we can obtain an image of two orthogonal sets of parallel coplanar space lines and the length of two unparallel line segments in the plane which is called reference plane, then we can uniquely determine the metric distance between any two points on the reference plane.

Define a space coordinate system XYZ, and let the origin of the coordinate frame lie on the reference plane, with X and Y axes parallel to the two sets of parallel lines respectively. Denote the vanishing points of the parallel lines in X and Y directions as $\mathbf{v}_X$ and $\mathbf{v}_Y$ respectively, it is clear that $\mathbf{v}_X = s_1\mathbf{p}_1$, $\mathbf{v}_Y = s_2\mathbf{p}_2$ and the vanishing line $\mathbf{v}_L = \mathbf{v}_X \times \mathbf{v}_Y$, where, s_1 and s_2 are nonzero scales. As shown in section 2.1, there exists a homography between the reference plane and the image. So we can select the first two column of $\mathbf{H}$ as $\mathbf{v}_X$ and $\mathbf{v}_Y$. The homography should be of rank three, otherwise, the mapping from reference plane to image is degenerate. Therefore, the final column of $\mathbf{H}$ must not lie on the vanishing line. We can select the last column $\mathbf{h}_3$ as any vector that is linearly independent to $\mathbf{v}_X$ and $\mathbf{v}_Y$, such as $\mathbf{h}_3 = \mathbf{v}_L$. Through the constructed homography $\mathbf{H} = [\mathbf{v}_X,\mathbf{v}_Y,\mathbf{h}_3]$, an image point $\tilde{\mathbf{m}}$ can be mapped to a point $\tilde{\mathbf{x}}_H$ in an affine space, we call it H space in the following.

Proposition 1: The distance between any two points in the Euclidean space and the corresponding distance in H space are equal up to two common scales.

Proof: From equation (2) and the above analysis, we have

$$\begin{cases} \tilde{\mathbf{m}} = \rho_1\mathbf{P}_0\tilde{\mathbf{x}} = \rho_1[\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_4]\tilde{\mathbf{x}} \\ \tilde{\mathbf{m}} = \rho_2\mathbf{H}\tilde{\mathbf{x}}_H = \rho_2[s_1\mathbf{p}_1,s_2\mathbf{p}_2,\mathbf{h}_3]\tilde{\mathbf{x}}_H \end{cases} \quad (3)$$

where, $\tilde{\mathbf{m}}$, $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}_H$ are in normalized homogeneous form with $\tilde{\mathbf{m}} = (u,v,1)^T$, $\tilde{\mathbf{x}} = (X,Y,1)^T$, $\tilde{\mathbf{x}}_H = (X_H,Y_H,1)^T$, $\mathbf{P}_0 = [\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_4]$ and ρ_1, ρ_2 are nonzero scales. Eliminate $\tilde{\mathbf{m}}$ and expand the above equation, we have

$$\begin{aligned} \tilde{\mathbf{x}} &= \frac{\rho_2}{\rho_1}\mathbf{P}_0^{-1}\mathbf{H}\tilde{\mathbf{x}}_H = \frac{\rho_2}{\rho_1}[\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_4]^{-1}[s_1\mathbf{p}_1,s_2\mathbf{p}_2,\mathbf{h}_3]\tilde{\mathbf{x}}_H \\ &= \frac{\rho_2}{\rho_1}\begin{bmatrix} s_1 & 0 & a_1 \\ 0 & s_2 & a_2 \\ 0 & 0 & a_3 \end{bmatrix}\begin{bmatrix} X_H \\ Y_H \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \lambda_1 & 0 & k_1 \\ 0 & \lambda_2 & k_2 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}}\begin{bmatrix} X_H \\ Y_H \\ 1 \end{bmatrix} = \mathbf{A}\tilde{\mathbf{x}}_H \end{aligned} \quad (4)$$

where, $a_1 = \frac{|\mathbf{h}_3 \cdot \mathbf{p}_2 \cdot \mathbf{p}_4|}{|\mathbf{p}_1 \cdot \mathbf{p}_2 \cdot \mathbf{p}_4|}$, $a_2 = \frac{|\mathbf{p}_1 \cdot \mathbf{h}_3 \cdot \mathbf{p}_4|}{|\mathbf{p}_1 \cdot \mathbf{p}_2 \cdot \mathbf{p}_4|}$, $a_3 = \frac{|\mathbf{p}_1 \cdot \mathbf{p}_2 \cdot \mathbf{h}_3|}{|\mathbf{p}_1 \cdot \mathbf{p}_2 \cdot \mathbf{p}_4|}$, $\lambda_1 = \rho_2 s_1/\rho_1$, $\lambda_2 = \rho_2 s_2/\rho_1$, $k_1 = \rho_2 a_1/\rho_1$, $k_2 = \rho_2 a_2/\rho_1$ and $\rho_2 a_3/\rho_1 = 1$. We can learn from equation (4) that λ_1, λ_2 are independent of any correspondence of $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}_H$. Thus, a point in the Euclidean space and its corresponding point in H space are related with an affine transformation $\mathbf{A}$. For two points $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ in the Euclidean space and their corresponding points $\tilde{\mathbf{x}}_{H1}$ and $\tilde{\mathbf{x}}_{H2}$ in H space

$$\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_2 = \begin{bmatrix} \lambda_1 X_{H1} + k_1 \\ \lambda_2 Y_{H1} + k_2 \\ 1 \end{bmatrix} - \begin{bmatrix} \lambda_1 X_{H2} + k_1 \\ \lambda_2 Y_{H2} + k_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_1 \Delta X_H \\ \lambda_2 \Delta Y_H \\ 0 \end{bmatrix} \quad (5)$$

Therefore, the distance between any two points in the Euclidean space is a function of the two common scales (λ_1, λ_2) and the coordinates difference of the computed two points in H space. If we know two unparallel reference line segments and their length d_1 and d_2, then following equations hold:

$$\begin{cases} d_1^2 = \lambda_1^2 \Delta X_{H1}^2 + \lambda_2^2 \Delta Y_{H1}^2 \\ d_2^2 = \lambda_1^2 \Delta X_{H2}^2 + \lambda_2^2 \Delta Y_{H2}^2 \end{cases} \quad (6)$$

After the recovery of λ_1 and λ_2, we can immediately obtain all the metric distance between any two points in the reference plane according to the following equation.

$$d = \sqrt{\lambda_1^2 \Delta X_H^2 + \lambda_2^2 \Delta Y_H^2} \quad (7)$$

Some remarks:

(i) In equation (4), the last column of $\mathbf{A}$ means a translation between the origin of the affine coordinate system and that of the Euclidean coordinate system, while λ_1 and λ_2 mean the two scalings between the X and Y axes of the two coordinate systems.

(ii) If each set of parallel lines are composed of two lines, then the two sets of orthogonal parallel lines form a rectangle. In this case, we have the following proposition.

Proposition 2: The ratio of λ_1 and λ_2 is equal to that of the rectangle's two sides.

Proof: Without loss of generality, we can denote the coordinates of intersections and the image of the rectangle as shown in Fig.2. Assume all the points in Fig.2 are in homogeneous coordinates, then we have

$$\begin{cases} \mathbf{o} = \mu_1 \mathbf{P}_0 \mathbf{O}, & \mathbf{a} = \mu_2 \mathbf{P}_0 \mathbf{A} \\ \mathbf{b} = \mu_3 \mathbf{P}_0 \mathbf{B}. & \mathbf{c} = \mu_4 \mathbf{P}_0 \mathbf{C} \end{cases} \quad (8)$$

where, $\mu_1, \mu_2, \mu_3, \mu_4$ are nonzero scales. Vanishing points $\mathbf{v}_X$ and $\mathbf{v}_Y$ can be computed from the image as follows.

$$\mathbf{v}_X = (\mathbf{o} \times \mathbf{a}) \times (\mathbf{b} \times \mathbf{c}) = \mu_1\mu_2\mu_3\mu_4 (\mathbf{P}_0\mathbf{O} \times \mathbf{P}_0\mathbf{A}) \times (\mathbf{P}_0\mathbf{B} \times \mathbf{P}_0\mathbf{C})$$
$$= \mu_1\mu_2\mu_3\mu_4 |\mathbf{P}_0| \mathbf{P}_0 (\mathbf{O} \times \mathbf{A}) \times (\mathbf{B} \times \mathbf{C}) = \mu_1\mu_2\mu_3\mu_4 a^2 b |\mathbf{P}_0| \mathbf{p}_1 = \gamma_1 \mathbf{p}_1$$

$$\mathbf{v}_Y = (\mathbf{o} \times \mathbf{c}) \times (\mathbf{a} \times \mathbf{b}) = \mu_1\mu_2\mu_3\mu_4 (\mathbf{P}_0\mathbf{O} \times \mathbf{P}_0\mathbf{C}) \times (\mathbf{P}_0\mathbf{A} \times \mathbf{P}_0\mathbf{B})$$
$$= \mu_1\mu_2\mu_3\mu_4 |\mathbf{P}_0| \mathbf{P}_0 (\mathbf{O} \times \mathbf{C}) \times (\mathbf{A} \times \mathbf{B}) = \mu_1\mu_2\mu_3\mu_4 ab^2 |\mathbf{P}_0| \mathbf{p}_2 = \gamma_2 \mathbf{p}_2$$

where, $\gamma_1 = \mu_1\mu_2\mu_3\mu_4 a^2 b |\mathbf{P}_0|$, $\gamma_2 = \mu_1\mu_2\mu_3\mu_4 ab^2 |\mathbf{P}_0|$, $\mathbf{p}_1$ and $\mathbf{p}_2$ are the first and second column of $\mathbf{P}_0$ respectively. Therefore, from the above equations and equation (5), it is easy to see that the ratio of λ_1 and λ_2 is equal to that of the two sides of the rectangle. i.e.

$$\lambda_1 : \lambda_2 = a : b \quad (9)$$

(iii) If the two sets of parallel lines form a square, then we have $\lambda_1 = \lambda_2$. In this case, the distance between any two points in the Euclidean space and the corresponding distance in H space are equal up to one common scale, and one reference distance is enough to determine the scale.

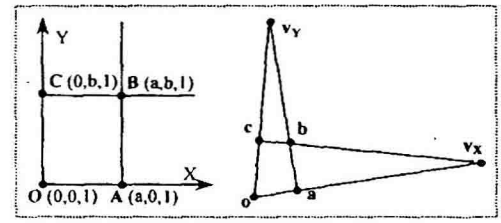

Fig.2 Two sets of parallel lines and their image

4. Experiments with simulated data

In this section, a comparative study of the proposed method as well as the homography based method is done via extensive simulations. In our simulations, the camera's setup is: $f_u = 1200$, $f_v = 1000$, $\alpha = 1$, $u_0 = 512$, $v_0 = 384$. The image resolution is: 1024×768 pixels. The extrinsic camera parameters are: rotation axes $r = [2, 1, 4]^T$, rotation angle $\alpha = \pi/6$ and translation $\mathbf{T} = [-5, -10, 800]^T$. We generate a rectangle in the reference space plane and evenly select 100 points on each side of the rectangle. The gaussian image noise (unit: pixel) is added on each image point, and the 100 image points on each side are fitted to a line via a least-square algorithm. In order to ensure the comparability of the two methods, all tests are taken under same camera parameters and simulation data.

For homography based method, first, we calculate the four corners of the generated rectangle by computing the intersections of the four fitted side lines and use these points to calculate the homography between the reference plane and image plane. Then an image point can be mapped to the Euclidean space via formula $\tilde{\mathbf{x}} = \mathbf{H}^{-1}\tilde{\mathbf{m}}$ and the distance between two space points can be determined.

For the proposed vanishing points based method, we take the rectangle as two sets of orthogonal parallel lines and take two unparallel line segments as the reference lengths so as to determine the two scales.

In order to provide more statistically meaningful results, we vary the added Gaussian noise from 0 to 5 pixels with a step of 0.1 pixel during the test. At each

noise level, we randomly select 200 pairs of space points and use their corresponding image points to estimate their metric distances by the two methods respectively. The result is shown in Fig.3. and Fig.4. In each figure, the left one is the relative error of the estimated distance at different noise level, while the right one shows their corresponding standard deviations.

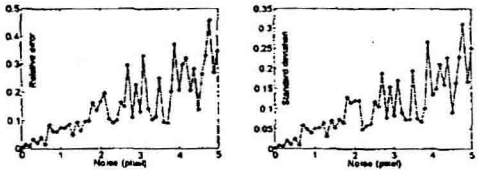

Fig.3 Test result of homography based method

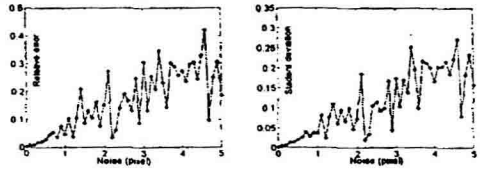

Fig.4 Test result of vanishing point based method

5. Experiments with real images

In real image test, images are taken by a digital CCD camera with resolution of 1024×768, then the canny edge detector and a least-squares technique are used to detect the edge points of the parallel lines and fit them into lines. Fig.5 shows two images of the test set.

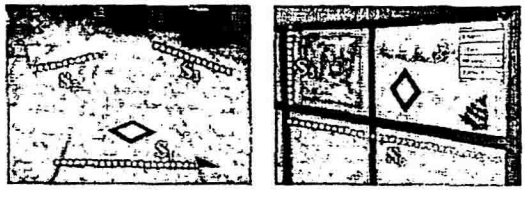

Fig.5 Two images of the test data

In order to detect the two orthogonal parallel lines as well as the control points and vanishing points automatically, we add a rectangle template to the scene. For homography based method, the metric measurements of the rectangle should be taken beforehand so as to calculate the homography. For vanishing points based algorithm, we just need to know two or one (for square case) unparallel reference lengths, which are not too demanding. Table 1 gives some comparative results of the test. In Table 1, HM stands for homography based method, VP for vanishing point based one, D_R for real distance, D_E for estimated distance and E_r for relative error.

Table 1 Real image test results Unit: cm

Line segments		S_1	S_2	S_3	S_4	S_5	S_6
D_R		60.05	84.85	134.18	53.65	53.65	189.78
HM	D_E	59.44	85.50	136.46	53.92	53.26	191.02
	E_r	1.02	0.77	1.7	0.5	0.73	0.65
VP	D_E	59.99	83.86	131.76	54.05	53.38	190.92
	E_r	0.1	1.17	1.8	0.75	0.5	0.6

6. Conclusions

In this paper, we propose a novel approach for single view based plane metrology. From the experiments with both simulated data and real images, we can learn that the method is of equal precision and robustness compared with the homography based one. It is worth noting that the method proposed here is easier to implement than others, especially in cases where control points are difficult to specify.

It is clear that the precision of this method depends greatly on that of the line fitting and vanishing point calculations. So, it is crucial to select a proper edge detecting and line fitting technique so as to increase the precision of distance measurements.

Acknowledgements: The work is supported by the National Grand Fundamental Research 973 Program of China under grant No. G1998030502 and the National Natural Science Foundation of China under grant No. 60033010 and grant No. 69975021.

References

[1]. A. Criminisi, Accurate visual metrology from single and multiple images, *Ph.D. thesis*, University of Oxford, 1999

[2]. A. Criminisi, I. Reid, and A. Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8): pp. 625~634, 1999.

[3]. D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. *Proc. of EuroGraphics*, Sep. 1999

[4]. T. Kim, Y. Seo, and K. Hong. Physics-based 3D position analysis of a soccer ball from monocular image sequences. *Proc. of ICCV*, pp. 721~726, 1998.

[5]. I. Reid and A. Zisserman. Goal-directed video metrology. In R. Cipolla and B. Buxton, editors. *Proc. of ECCV*, volume II, pp. 647~658. Springer, Apr. 1996

[6]. D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. *Proc. of CVPR*, pp. 482~488, Jun. 1998.

[7]. R. Hartley and A. Zisserman. Multiple view geometry in computer vision, *Cambridge University Press*, 2000

[8]. R. T. Collins and R. S. Weiss. Vanishing point calculation as a statistical inference on the unit sphere. *Proc. of ICCV*, pp. 400~403, Dec. 1990.

[9]. G. McLean and D. Kotturi. Vanishing point detection by line clustering. *IEEE T-PAMI*, 7(11) pp. 1090~1095, 1995