Analysis of Gesture and Action in Technical Talks for Video Indexing

Shanon X. Ju* and **Michael J. Black†** and **Scott Minneman†** and **Don Kimber†**
* Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4 Canada
† Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304
juxuan@vis.toronto.edu, {black,minneman,kimber}@parc.xerox.com

Abstract

In this paper, we present an automatic system for analyzing and annotating video sequences of technical talks. Our method uses a robust motion estimation technique to detect key frames and segment the video sequence into subsequences containing a single overhead slide. The subsequences are stabilized to remove motion that occurs when the speaker adjusts their slides. Any changes remaining between frames in the stabilized sequences may be due to speaker gestures such as pointing or writing and we use active contours to automatically track these potential gestures. Given the constrained domain we define a simple "vocabulary" of actions which can easily be recognized based on the active contour shape and motion. The recognized actions provide a rich annotation of the sequence that can be used to access a condensed version of the talk from a web page.

Introduction

In recent years, researchers have been increasingly interested in the problem of browsing and indexing video sequences. The majority of work has focused on the detection of key frames and scene breaks in general, unconstrained, video databases (Otsuji & Tonomura 1994; Zabih, Miller, & Mai 1996; Zhang, Kankanhalli, & Smoliar 1993). For these methods to work on general video sequences they use simple image processing techniques and do not attempt any high-level analysis of the content of the sequence. In our work we have chosen to constrain the domain of video sequences that we wish to analyze and look specifically at video-taped presentations in which the camera is focused on the speaker's slides projected by an overhead projector. By constraining the domain we are able to define a rich "vocabulary" of actions that people perform during a presentation. By automatically recognizing these actions we can provide a rich annotation of the video sequence that can be used, for example, to access a summarized or condensed version of the talk from a web page.

Figure 1 shows a simple example of a video browsing and indexing system. The original video stream is summarized and annotated first. In previous versions of the system this was an off-line process and the present paper addresses the automation of this process. The outputs of the process are

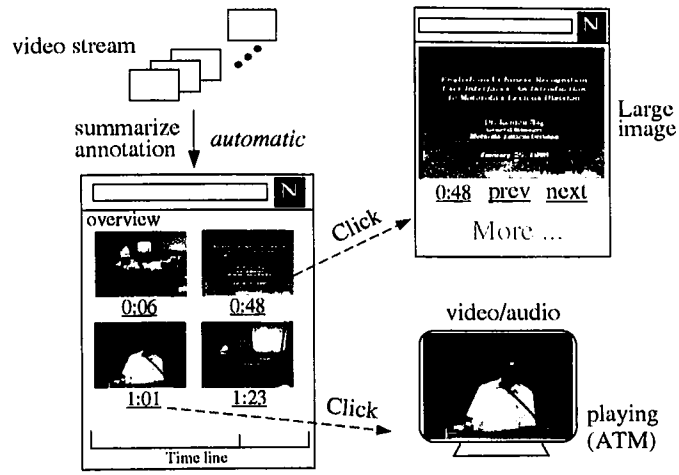


Figure 1: A example system of video browsing and indexing

indices of the events and the images corresponding to these events. This information is used to make a summary web page containing images of each event and their time indices. Our current system makes use of real-time JPEG decoding hardware and a fiber-optic ATM network to permit access of real-time audio and video from the web page.

Generally speaking, the goal of automatic video annotation is to save a small set of frames that contain most of the relevant information in the video sequence. In our restricted domain of overhead presentations a number of "changes" can occur in the image sequence. There are two classes which we will call "nuisances" and "affordances"[1]

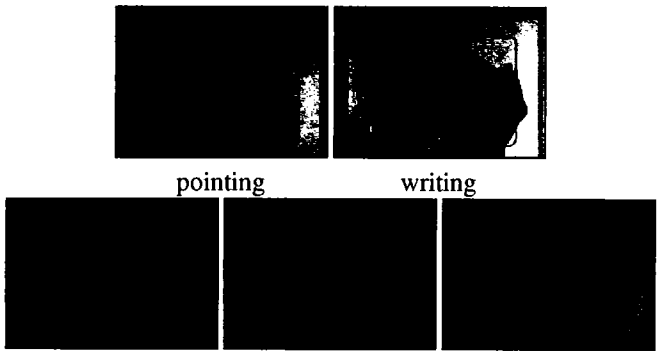
Nuisance changes are those which we define to have no relevant semantic interpretation (Figure 2). Examples of this are when the speaker occludes the slide with their hand or body or when the speaker moves the slide (an action that we observe to be very common). These nuisance changes are ones that we wish to ignore in the analysis and summarization of the video sequence.

Affordances, on the other hand, are changes in the video

[1] The actual distinction between these classes is somewhat arbitrary and is defined relative to the specific domain formulation and problem to be solved.

occlusion motion

Figure 2: Nuisance changes.



pointing writing

revealing

Figure 3: Affordances.

sequence that have a semantic interpretation with respect to the presentation (Figure 3). For example, speakers often write, point, or make repetitive gestures at locations on the slide to which they are referring. Another common action is to cover a portion of the slide and gradually reveal the underlying text. We call these changes "affordances" because we can take advantage of them to acquire more information about the presentation.[2] Automatic recognition of the affordances can provide a rich description of the video. As we will show, recognition of the affordances will allow us to produce annotated key-frames from the video that will allow users to later access portions of the talk where the speaker gestured at a particular location on his or her slides.

In this paper we propose a novel approach of automatic meeting video annotation. First, we estimate the global image motion between every two consecutive frames using a robust regression method. The motion information is used to compute a warped sequence where the slides are stabilized. Second, the stabilized sequence is processed to extract *slide templates*, or key frames. Third, we compute a pixel-wise difference image between the slide templates and the corresponding frames in the stabilized image sequence. These difference images contain only occluded/disoccluded objects, e.g., the gestures. Then we track these gestures using a deformable contour model. By analyzing the shape of the contour and its motion over time, we can recognize pointing gestures and recover the location on the slide to which the speaker is referring. Finally, the key frames and gesture information can be integrated to annotate the video. We will describe the first three steps in Sections 3 through 6. In Section 2, we briefly review previous work on video summarization.

[2] Gibson (Gibson 1979) uses the term *affordances* to describe invariant properties of an environment that an organism can use to their advantage.

Related Work

The two main themes explored in previous work on automatic video summarization can be broadly described as *segmentation* and *analysis*. The work on segmentation focuses on finding scene changes or key frames in the video while work on analysis focuses on understanding actions or events (typically in a more restricted domain). Both types of analysis are important to provide useful video summarization.

Scene-break detection is a first step towards the automatic annotation of digital video sequences. There are two basic types of algorithms for scene-break detection. The first uses image-based methods, such as image differencing and color histograming (Otsuji & Tonomura 1994; Zhang, Kankanhalli, & Smoliar 1993). The second, feature-based methods, use image edge pixels (Zabih, Miller, & Mai 1996). These algorithms typically compute the differences between two consecutive images and, when the difference is larger than a threshold, there may be a scene break.

Simple image-based differencing tends to over-segment the video sequence when there is motion present in the scene or when the camera is moving since many pixels will change their color from frame to frame. Zabih *et al.* (Zabih, Miller, & Mai 1996) recently proposed a feature-based method. They detected the appearance of intensity edges that are distant from edges in the previous frame. A global motion computation is used to handle camera or object motion. Their method can detect and classify scene breaks that are difficult to detect with image-based methods. However their motion estimation technique (the correlation method and the Hausdorff distance method) can not handle multiple moving objects well. Generally speaking, the image- and feature-based methods are naive approaches that use straightforward measurements of scene change. For simply detecting scene changes, these methods are fast.

There has been somewhat less attention paid to the semantic analysis of video and arguably this is a tremendously hard problem in general and previous work has focused on narrow domains. Intille and Bobick (Intille & Bobick 1995) refer to these restricted domains as "closed-worlds" and their work focuses on the tracking and analysis of players in a football game. As in our work, they must register multiple frames in a video sequence in which other changes are occurring. They manually register the frames while we use an automatic robust estimation technique to perform this task. Their domain did not require them to address the segmentation of the video into multiple scenes (cut detection). Like us, they define a narrow domain in which they can ex-

press prior assumptions about the semantics of changes in the scene.

Other recent attempts to provide an automated analysis of video in restricted domains include the work of Mann *et al.* (Mann, Jepson, & Siskind 1996) who propose a method for analyzing the physical interactions between objects in a video sequence and that of Brand (Brand 1996) who looks at understanding human actions in video for the purpose of video summarization. Earlier work on the linguistic analysis of action in video focused on generating descriptions of a soccer match given manually generated motion information (Andre, Gergog, & Rist 1988; Retz-schmidt 1988).

We will present a robust motion estimation method in the following section, which can recover the motion of overhead slides accurately. This will be used to automatically stabilize the image sequence and to reliably detect slide changes in the sequence. We then present a method for gesture tracking and recognition.

Motion estimation

Image motion between two frames can be estimated using a parametric model. Parametric models of image motion make explicit assumptions that the image flow can be represented by a low-order polynomial. For example, we assume that the slides are always perpendicular to camera's viewing axis, and they can be modeled as a rigid plane. Therefore, the image motion can only be translation, scaling, or rotation. For small motions, these can be described by the following four-parameter model:

$$u(x,y) = a_0 + a_1 x - a_2 y, \quad (1)$$

$$v(x,y) = a_3 + a_2 x + a_1 y, \quad (2)$$

where $\mathbf{a} = [a_0, a_1, a_2, a_3]$ denotes the vector of parameters to be estimated, and $\mathbf{u}(\mathbf{x}, \mathbf{a}) = [u(x,y), v(x,y)]$ are the horizontal and vertical components of the flow at image point $\mathbf{x} = [x, y]$. The coordinates (x, y) are defined with respect to a particular point; here this is taken to be the center of the image.

To estimate the motion parameters, $\mathbf{a}$, for a given patch we make the assumption that the brightness pattern within the patch remains constant while the patch may translate, scale, or rotate. This brightness constancy assumption is formulated as

$$I(\mathbf{x} + \mathbf{u}(\mathbf{x}, \mathbf{a}), t+1) = I(\mathbf{x}, t), \ \forall \mathbf{x} \in \mathcal{R} \quad (3)$$

where $\mathbf{a}$ denotes the motion model for patch $\mathcal{R}$, I is the image brightness function and t represents time. This equation simply states that the motion $\mathbf{u}(\mathbf{x}, \mathbf{a})$ can be used to warp image at time $t+1$ to make it look like the image at time t.

Note that the brightness constancy assumption is often violated in practice due to changes in lighting, occlusion boundaries, specular reflections, etc. In our domain of view-graphs, violations will occur in situations in which the speaker occludes their slides. Robust regression has been shown to provide accurate motion estimates in a variety of situations in which the brightness constancy assumption is violated (Black & Anandan 1996). To estimate the slide motion, $\mathbf{a}$, robustly, we minimize

$$E_s = \sum_{\mathbf{x} \in \mathcal{R}} \rho(I(\mathbf{x} + \mathbf{u}(\mathbf{x}, \mathbf{a}), t+1) - I(\mathbf{x}, t), \sigma), \quad (4)$$

with respect to the parameters $\mathbf{a}$ for some robust error norm ρ where σ is a scale parameter (see (Black & Anandan 1996) for details). Violations of the brightness constancy assumption can be viewed as "outliers" (Hampel *et al.* 1986) and we need to choose the function ρ such that it is insensitive to these large errors.

Equation 4 is minimized using a simple gradient descent scheme (Black & Anandan 1996). The robust formulation of Equation 4 means that the algorithm estimates the dominant motion in the scene (i.e. the slide motion) and automatically ignores the image points that belong to other motions (the gestures). Gestures will be tracked using a different technique described in the Section 5. In the following section we will use the motion information to *stabilize* the image sequence with respect to *key frames*.

Key-Frame Detection

Given an image sequence corresponding to a particular slide, stabilization is just a process of *warping* each of the images towards a *reference image* by taking into account the cumulative motion estimated between each of the frames in the sequence. Since minimizing Equation 4 can only estimate small image motions between two consecutive images, we need to compute the motion between the *reference image* and each following frame. Given $\mathbf{a}_{n-1}$, the motion between the *reference image* and frame $n-1$, and $\mathbf{a}_n^*$, the motion between frame $n-1$ and n, image motion between the *reference image* and frame n is:

$$\mathbf{a}_n = \mathbf{a}_{n-1} + \mathbf{a}_n^* + \mathrm{d}\mathbf{a}$$

$$\mathrm{d}\mathbf{a} = [a_{n_1}^* \, a_{n_2}^*] \begin{bmatrix} a_{n-1_0} & a_{n-1_1} & a_{n-1_2} & a_{n-1_3} \\ a_{n-1_3} & -a_{n-1_2} & a_{n-1_1} & -a_{n-1_0} \end{bmatrix}$$

where a_{n-1_3}, for example, represents the paramter a_3 from the previous frame $n-1$.

We use a simple heuristic that the reference frame is the first non-blank image for which the motion is smaller than a threshold. For subsequent images, if the motion estimation method succeeds (has low error) then the image belongs to the same subsequence as the reference frame. When the motion estimation method fails, it means that two consecutive frames are significantly dissimilar and can not be modeled by Equations 1 and 2. Thus, it typically corresponds to a change of slides, and we use this frame to end the subsequence and we begin looking for the next stable reference

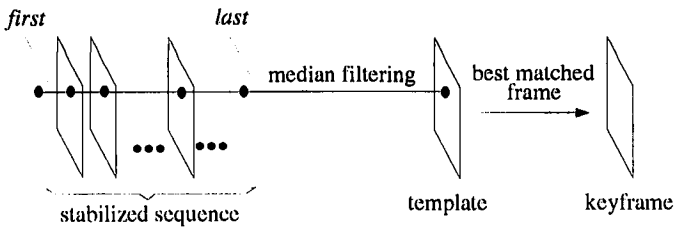


Figure 4: The key frame detection: filtering the stabilized sequence.

image. In the domain of overhead presentations this simple strategy works well.

Since we only estimate the dominant (slide) motion in the scene, the warped sequence contains both the stabilized slide and moving objects, such as the hand of the speaker. To get a template image that contains no gestures, we use a median temporal filter to remove the moving objects in the sequence (Wang & Adelson 1993). At each image position, we take all the values at this position in the stabilized frames, find the median value, and use it as the intensity value of the slide template. The median filter can filter out the gestures, hand motions, and partial occlusion. Finally we find which of the stabilized frames is most similar to the template, and use this particular frame as our key frame of the slide (see Figure 4).

Experimental results

We collected a short image sequence to simulate a talk (approximately 3200 frames over 105 seconds of video) in which the camera is focused on the speaker's slides on a desktop. The speaker used three slides during the talk. They put the first slide on, moved it up then down, pointed to a few bullets on the slide, and then took it away. The second slide was partly covered at first, then the cover sheet was moved down to reveal the slide. The speaker also wrote a few words on the third slide. During the "talk", the speaker frequently adjusted the position of their slides.

Figure 5 shows the three automatically recovered key frames. Since we warp all the frames backward toward the first reference frame, the slides should be put roughly in the center of the viewing area at first. Note that, while it did not occur with this sequence, the median filter could produce unexpected results. For instance, if the speaker puts their hand on the slide for a significantly long period of time, the hand will be considered as part of the key frame or template. More sophisticated techniques would be required to distinguish hands from slides but it is not clear whether this is necessary or desirable.

Gesture Tracking

If we compute the absolute difference between the slide template and images in the warped sequence, the non-zero

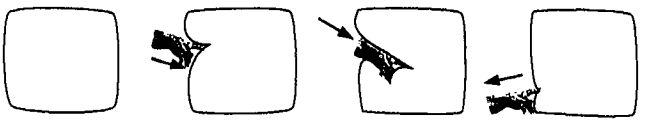


Figure 6: Gesture Tracking: a deformable image boundary.

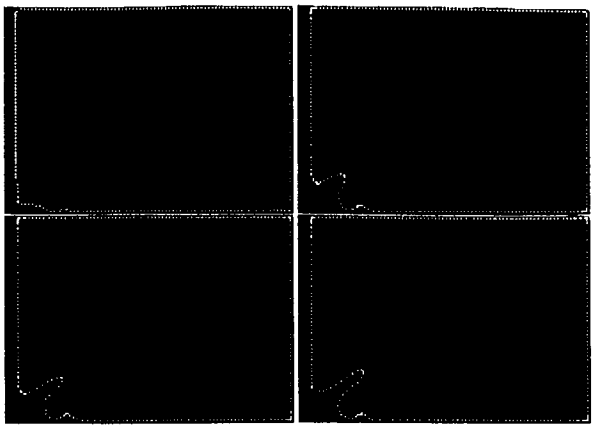


Figure 7: The snake at iteration 1, 20, 40 and 60.

pixels in the image must correspond to gestures, covered data, or written text. Since all the gestures must enter the scene or leave the scene from the image boundary, new material can not suddenly appear in the middle of the image. Therefore, we can let the image boundary deform when a "thing" enters, such that it tracks the contour of the entering object. If the object leaves, the deformable contour will expand to the image boundary (See Figure 6).

We use controlled continuity splines, or "Snakes" (Kass, Witkin, & Terzopoulos 1987), to model the deformable contour. The idea is to have the snake lock on to features of an image structure by minimizing an integral measure which represents the snake's total energy. Due to the dynamic property of the snake model, we can achieve automatic tracking of a contour from frame to frame.

The behavior of a snake is controlled by internal and external forces. The internal forces serve as a smoothness constraint, and the external forces guide the active contour towards image features. Following the notation from the original model proposed by Kass *et al.* (Kass, Witkin, & Terzopoulos 1987), given a parametric representation of an image curve $\mathbf{v}(s) = (x(s), y(s))$, the energy function is defined as

$$\mathcal{E}_{snake} = \int_0^1 \mathcal{E}_{int}(\mathbf{v}(s)) + \mathcal{E}_{ext}(\mathbf{v}(s))ds. \qquad (5)$$

The function $\mathcal{E}_{int}$ represents the internal energy of the active contour and is composed of a first and second order derivative terms ($\mathbf{v}_s$ and $\mathbf{v}_{ss}$ respectively):

$$\mathcal{E}_{int} = (\alpha|\mathbf{v}_s(s)|^2 + \beta|\mathbf{v}_{ss}(s)|^2)/2. \qquad (6)$$

(a) (b) (c)

Figure 5: The key frames of: (a) slide 1 (b) slide 2 (c) slide 3

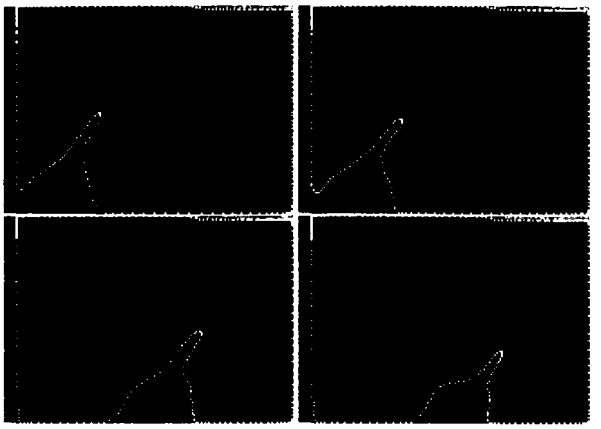


Figure 8: Snake Tracking at frame 1220, 1240, 1280 and 1312.

The first-order term makes the snake act like a string and the second-order term makes it act like a rod. Adjusting the weights α and β controls the relative importance of the first and second terms.

$\mathcal{E}_{ext}$ represents the external potential

$$\mathcal{E}_{ext} = P(x,y) - K = c[G_\sigma * \Psi(x,y)] - K,$$

where K is a constant expansion force, c is a constant weight, Ψ is a difference image, and $G_\sigma * \Psi$ denotes the difference image convolved with a Gaussian smoothing filter. The active contour model in Equation (5) attempts to find a contour which is both smooth and which minimizes the value of $P(x,y)$ at every snake node (x,y). $P(x,y)$ is a scalar potential function defined over image plane. If the value of P is large over some region that overlaps the boundary of the image (because the hand has entered the slide) then the snake will deform around the region until $P(x,y)$ is small enough along the contour and both the internal and external forces are balanced. If the hand leaves the frame the default expansion force will push the snake back out to the image boundary.

Minimizing the energy function of Equation 5 gives rise to two independent Euler equations (Kass, Witkin, & Terzopoulos 1987). The tracking behavior of the snake is

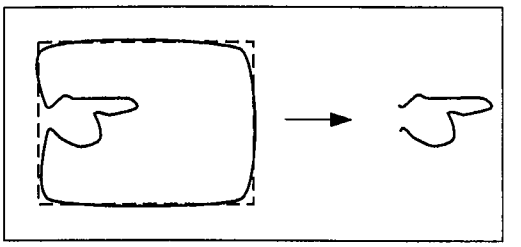


Figure 9: Finding the contour of the object.

achieved by numerical, iterative solution of these two equations using techniques from variational calculus (See (Kass, Witkin, & Terzopoulos 1987) for details).

It is well known that if the initial snake position is not chosen properly, the contour will fail to converge to desirable features. We can initialize a closed snake to be at the position of image boundary, and avoid the hard problem of automatic initialization of snakes. Figure 7 shows how the snake deforms to find the object boundary. The image shows the absolute difference between one of the frames in the sequence and the corresponding key frame. Bright areas correspond to the hand of the speaker making a pointing gesture. Once the snake locks onto image features, we can change the external image to the next frame. The snake is able to track the moving object when the images change (See Figure 8). In Figure 7 and 8, white dots indicate the snake nodes.

There is a problem of using the difference images for the external potential. If the entering object has a color similar to that of the slide, the snake will not track it since the external image force will be too small. In our experiments, the snake model failed to track a gesture sequence involving pointing with a pen which was relatively small and of similar color to the slide.

Recognizing Pointing Gestures

From the snake nodes, we can detect if, and where, the speaker is pointing on the slides. Our method has three steps. First, we fit a bounding quadrilateral to the snake nodes by finding the four corners. The nodes that are not close to any edge of the quadrilateral belong to the deformed

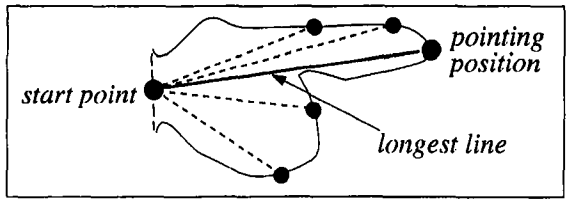


Figure 10: Pointing position

part of the Snake, that is, the contour of the object (Figure 9).

Second, we define a starting point to be the middle point between the first node and the last node of the contour (Figure 10). Among all the snake nodes on the contour, the one that is furthest from this starting point is defined as the pointing position. The line which connects the starting point and the pointing position will give us the rough pointing direction.

Finally, we will recognize the pointing gesture. Two heuristics are used to filter out the non-gestures. The first is a constraint on the spatial structure of the gesturing object; we want it to have a definite "point". The pointing positions that are too close to the first or last node in the segmented gesture are therefore eliminated (Figure 11, row one).

The second heuristic models the expected temporal pattern of gestures. We only recognize an action as a pointing gesture if the speaker points at one position for a time longer than a threshold (e.g., longer than 0.5 second). In the second row of Figure 11, a hand moves continuously from left to right in the sequence. We will not classify this gesture as pointing. In the third row, on the other hand, the finger stays at roughly the same position for a while, and it will be recognized as a pointing gesture. More sophisticated techniques could be employed for recognizing more complex gestures (cf. (Bobick & Wilson 1996)).

On the first slide of our experimental sequence, the speaker pointed to the two top boxes and the two following bullets. Figure 12 shows a few images from this portion of the experimental sequence. During this portion of the talk the speaker moved the slides a number of times.

Figure 13 shows the resulting analysis of our system for this portion of the sequence. Four distinct pointing gestures were recognized and the estimated pointing positions are marked on the slide template by a pointing-hand icon. These four slides become part of the web-based interface to the talk allowing the user to access the original video and audio at those points in the talk relevant to their interests. Note that nuisance gestures corresponding to the speaker moving the slides have been correctly eliminated.

Conclusion

We propose a fully automatic method that can robustly detect key frames of a video captured in a meeting. The

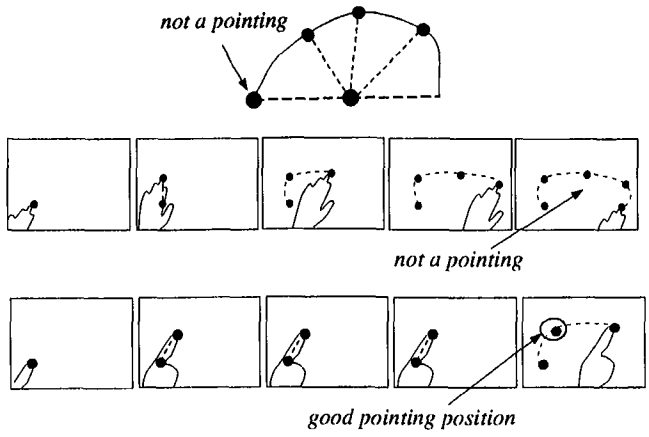


Figure 11: Good and bad pointing.

method is robust with respect to slide motions, occlusions and gestures. It can also provide richer description of the slides, such as where the speaker is pointing. This automatic video annotation and analysis system will help the user access meeting videos intelligently.

Note that this is an off-line process. The original digitized video sequence is saved in JPEG format and must be decompressed before processing. The decompression, motion estimation and key-frame detection parts of our algorithm take approximately 20 minutes to process the near 2 minute video of our simulated talk. This time is highly dependent on whether or not there are a lot of changes in the scene. Since the sequence we use has a large number of motions and gestures in short period of time, the cost of motion estimation is high. With gesture tracking, the algorithm takes nearly 50 minutes in total for the 2 minute sequence.

In future work we would like to match the low resolution slide template image with a stored postscript file of the slides. This would provide an automatic correspondence between the low-resolution video and a postscript version of the talk and would allow a user to view or print specific slides at high resolution. We also leave the detection of revealing and writing affordances for future work.

Acknowledgements

We thank David Fleet for his comments on the manuscript.

References

Andre, E.; Gergog, G.; and Rist, T. 1988. On the simultaneous interpretation of real world image sequences and their natural language descriptions. In *European Conference on Aritificial Intelligence*, 449–454.

Black, M., and Anandan, P. 1996. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. In *Computer Vision and Image Understanding*, volume 63(1), 75–104.

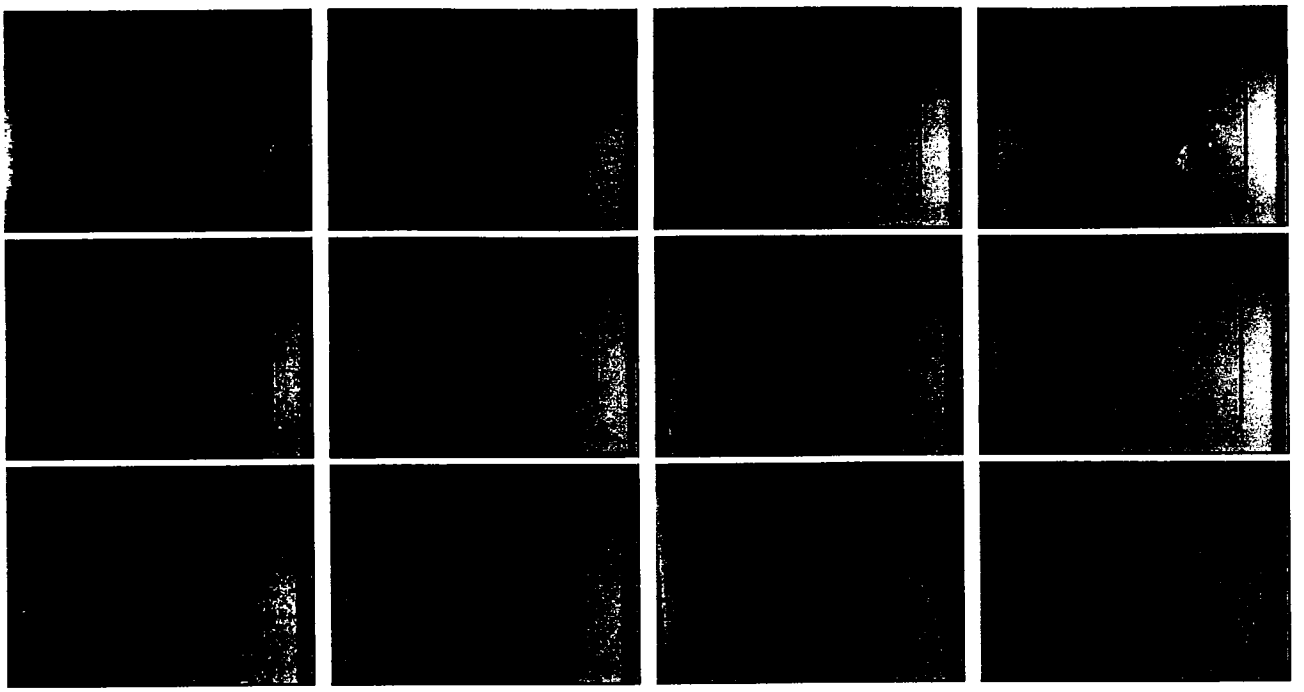


Figure 12: Example frames taken from the first slide sequence. Every 55th frame is shown.

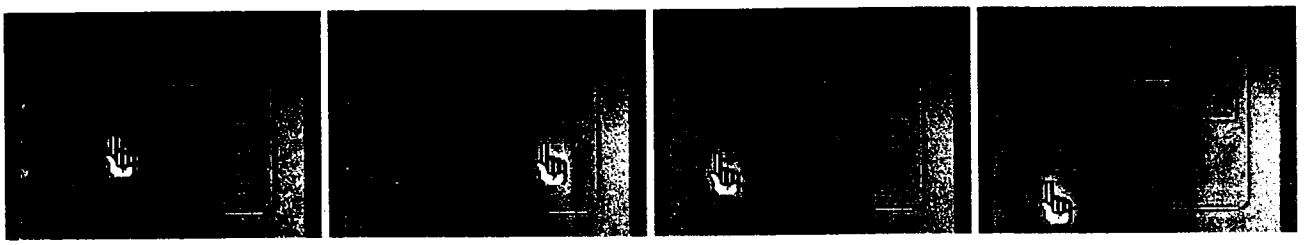


Figure 13: Recognized pointing gestures their positions (for the first slide).

Bobick, A. F., and Wilson, A. D. 1996. A state-based technique for the summarization and recognition of gesture. In *ICCV*, 382–388.

Brand, M. 1996. Understanding manipulation in video. In *International Conference on Automatic Face and Gesture Recognition*, 94–99.

Gibson, J. J. 1979. *The Ecological Approach to Visual Perception*. Boston, MA: Houghton Mifflin.

Hampel, F. R.; Ronchetti, E. M.; Rousseeuw, P. J.; and Stahel, W. A. 1986. *Robust Statistics: The Approach Based on Influence Functions*. J. Wiley & Sons.

Intille, S. S., and Bobick, A. F. 1995. Closed-world tracking. In *Proceedings of the International Conference on Computer Vision*, 672–678.

Kass, M.; Witkin, A.; and Terzopoulos, D. 1987. Snakes: Active contour models. In *ICCV*, 259–268.

Mann, R.; Jepson, A.; and Siskind, J. M. 1996. Computational perception of scene dynamics. In Buxton, B., and Cipolla, R., eds., *European Conf. on Computer Vision, ECCV-96*, volume 1064 of *LNCS-Series*, 528–539. Cambridge, UK: Springer-Verlag.

Otsuji, K., and Tonomura, Y. 1994. Projection-detecting filter for video cut detection. In *Multimedia Systems*, volume 1, 205–210.

Retz-schmidt, G. 1988. Recognizing intentions in the domain of soccer games. In *European Conference on Aritificial Intelligence*, 455–457.

Wang, J. W. A., and Adelson, E. H. 1993. Layered representation for motion analysis. In *CVPR*, 361–366.

Zabih, R.; Miller, J.; and Mai, K. 1996. Video browsing using edges and motion. In *CVPR*, 439–446.

Zhang, H.; Kankanhalli, A.; and Smoliar, S. W. 1993. Automatic partition of full-motion video. In *Multimedia Systems*, volume 1, 10–28.