An Evaluation of Optical Flow using Lucas and Kanade's Algorithm

Carman Neustaedter
University of Calgary
Department of Computer Science
Calgary, AB, T2N 1N4 Canada
+1 403 210-9507
carman@cpsc.ucalgary.ca

ABSTRACT
Video offers distance-separated co-workers with a rich awareness of who is available for conversation or collaboration. By broadcasting video, however, the privacy of individuals becomes threatened because others now see potentially sensitive information. This risk is heightened for telecommuters who use video to connect to a remote office from home. Blurring video is one technique that has previously been studied to help preserve privacy. Most blurring techniques blur the entire video, however, it is desirable to be able to blur regions of the video differently if multiple people are present. Optical flow can be used in this situation to distinguish which regions contain different people by their level of activity. This paper evaluates an implementation of Lucas and Kanade's algorithm for computing optical flow and discusses possible applications for it in videoconferencing that is sensitive to privacy issues. The implementation is found to compute accurate optical flow for small pixel displacements, yet real-time computation is only possible for small frame sizes.

Keywords. Optical flow, Lucas and Kanade, video conferencing, motion, privacy.

INTRODUCTION
Everyday communication and interaction between co-workers is held together by an awareness of those who are around and available [2,5,9]. Awareness helps people decide if and when to smoothly move into and out of conversation or collaboration. This awareness is easily and naturally gained by those located in close physical proximity [7,16]. People can see if another is busy by simply glancing in their office or cubicle. As people become spread over distance, however, this awareness becomes difficult to gain unless technology is present [7,10].

Video is capable of providing rich awareness over distance and has been suggested by many researchers as a technology capable of mediating communication [2,4-8,11,14,15,17]. As video broadcasts information about one's awareness, such as a user's current activity and level of availability, it may be presenting information at an undesirable level and threatening the user's privacy however. The user's privacy could be violated just as easily though if no awareness information is presented because a co-worker could interrupt at an inappropriate time.

Telecommuters who work from home and connect to a remote office using video face increased privacy threats, as homes are inherently private in nature. Video links place telecommuters in two different environments where there is a disparity between levels of appropriateness. For example, working without a shirt on may be quite appropriate for those working at home, but the same level of undress is not appropriate for an office environment. Video attempts to place the telecommuter under the appropriateness constraints of both locations. To further complicate the situation, others present in the home, such as family members who gain no benefit from the video link still incur the privacy threat.

Blur filtration is one technique that has been studied to mask sensitive details in video that may be considered threatening to one's privacy [5,17], such as a person's activity, a person's appearance, or the location and its appearance. Most filtration techniques are only capable of masking the entire video sequence an equal amount. In cases where multiple people are present within a home there exists a desire to be able to blur regions of the video differently for each person. Some people, such as the telecommuter working, may be appropriate to broadcast at full fidelity while someone walking into the room in varying levels of undress may not be. The person working could be blurred less than the person walking into the room. For such situations, being able to distinguish between a high level of activity and a low level of activity would be desirable. Areas of high activity, such as a secondary person walking into a room could be blurred more than a region containing someone working.

Optical flow algorithms show potential in this area because they are able to compute motion between frames of video. The optical flow that is computed for each pixel in a frame could be used to determine which regions of the video frame contain large amounts of flow or motion. These regions could then be treated accordingly for their suspected level of privacy threat. Many optical flow algorithms are quite computationally expensive, however, and not suitable for use while transmitting video to others. Lucas and Kanade [13] present a differentiation method for computing optical flow that has shown computational

potential in various implementations by other researchers [1,12].

This paper discusses an implementation of the Lucas and Kanade algorithm for computing optical flow and its application to privacy preservation techniques for mediating privacy when video is used in homes. The implementation is evaluated based on its accuracy in computing flow and distinguishing levels of high activity from levels of low activity. The computation performance of the implementation is also analyzed for various frame sizes.

RELATED WORK

The algorithm presented by Lucas and Kanade [13] is an image registration technique that can be used to compute optical flow. Image registration techniques attempt to find an optimal value for a disparity vector, h, which represents an object's displacement between successive images. Methods to find the disparity vector prior to Lucas and Kanade relied on computationally expensive or non-optimal techniques. One such method performs an exhaustive search for h that examines all possible values for the disparity vector [13]. An alternative strategy is found in hill-climbing techniques that evaluate difference functions at all points in a region, choosing the point that minimizes the difference [13]. Hill-climbing techniques can lead to false peaks that produce local, rather than global, optimums for h however. A third technique, sequential similarity detection, computes an error value for each possible disparity vector and then applies a method similar to alpha-beta pruning in min-max trees, where disparity vectors with large error values are eliminated [13].

Lucas and Kanade's algorithm [13] presents an image registration method that examines fewer possible values of h than previous algorithms to improve computation performance. An initial estimate of h is made and then iteratively updated based on the average spatial intensity gradient found at each point within a fixed sized window. The contribution of each gradient to the value of h is weighted based on an estimate of the gradient's error. Iteration ceases after a specified length of time or when h converges. Smoothing an image is shown to help improve convergence, but can cause greater errors if an object is suppressed entirely [13]. It is assumed in the algorithm that pixels from the image are moving at a constant velocity and corresponding pixels in the images fall within a fixed sized window. An implemented version of Lucas and Kanade's algorithm is described in Barron, Fleet, and Beauchemin [1].

$$\begin{bmatrix} \Sigma\, wI_x^2 & \Sigma\, wI_xI_y \\ \Sigma\, wI_xI_y & \Sigma\, wI_y^2 \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = - \begin{bmatrix} \Sigma\, wI_xI_t \\ \Sigma\, wI_yI_t \end{bmatrix}$$

Figure 1: The linear equation used for computing the optical flow components, *u* and *v*

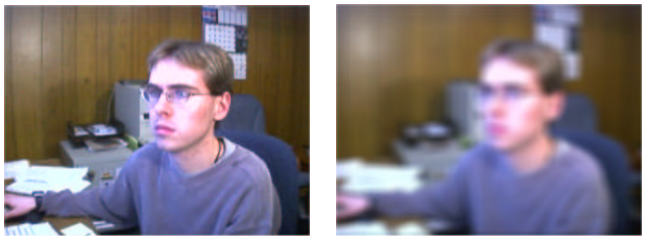


Figure 2: A video frame (left) compared to its smoothed version (right).

Lim and Gamal [12] use a modified version of Lucas and Kanade's algorithm that utilizes advancements in CMOS image sensor technology to compute more accurate optical flow measurements using high frame rates. Their technique first computes an optical flow estimate between two frames at high frame rates using the Lucas and Kanade method. Next, the optical flow is computed from the successive images at standard frame rates. This new value is combined with the initial estimate to create a final flow value. Lim and Gamal conclude that by using high frame rate sequences they are able to handle pixel displacements of up to 10 pixels/frame at 30 frames/sec, which is much higher than the reported 1-2 pixels/frame displacement capabilities of the original Lucas and Kanade algorithm.

IMPLEMENTATION

The Lucas and Kanade algorithm as described by Barron et al. [1] follows four main steps to compute optical flow for each frame in a video sequence. Frames are first smoothed with a spatiotemporal filter to reduce errors in gradient estimation (Figure 2). Next, gradients are estimated in *x*, *y*, and *t* (time) for each pixel in the current frame. Gradients are then smoothed and placed in a linear system (Figure 1). Finally, the linear system is solved for the two velocity components, *u* and *v*. This section describes the implementation used and discusses areas where deviations from other implementations were made.

Spatiotemporal Smoothing. Each frame must be smoothed to reduce errors that may be produced during differentiation. Smoothing first occurs spatially where a 1d filter is used to convolve the image first in *x* and then *y*, similar to standard image smoothing techniques. To smooth temporally, Barron et al. [1] suggest using a Gaussian filter with standard deviation of 1.5 pixels-frames. This requires the storage of a large set of frames for smoothing and delays the algorithm's flow output, which is undesired if the implementation is to be used in broadcasting video over the Internet. To alleviate this problem, my implementation uses a simple two-frame weighted average. The current pixel's value is averaged with the pixel from the same location in the previous frame, where α gives more weight to the value of the current

$$E(t = n) = (1-\alpha)\, E(n-1) + \alpha\, E(n)$$
$$\alpha = 0.75$$

Figure 3: The formula for temporally smoothing the value, *E*, of a pixel.

frame's pixel (Figure 3). A sample frame and its smoothed frame are shown in Figure 2.

Estimating the Gradients. Each frame is convolved in x, y, and t with a 1 x 5 kernel to compute I_x, I_y, and I_t for each pixel. The kernel used in the implementation is shown in figures 4 and 5 along with calculations of I_x and I_y using a sample set of image values. The same kernel is used by Barron et. al [1]. Gradients in x and y can be computed within the current frame, however I_t must be computed using a set of five frames – the current frame, two previous frames, and two subsequent frames – as shown in figure 6. The five-frame window needed for I_t causes output to be delayed by two frames, but for video conferencing this would be seemingly unnoticeable at standard video frame rates. As well, for the first two frames and the last two frames in a frame sequence optical flow is not computable because a five-frame window is not available. Next, I_x, I_y, and I_t are used to calculate I_x^2, I_y^2, I_xI_y, I_xI_t, *and* I_yI_t.

Gradient Smoothing. Gradients are now smoothed for each pixel in the frame within a 5 x 5 window using a 5 x 5 kernel. The kernel used, as suggested by Barron et. al [1], is shown in figure 7 along with a sample smoothing calculation for I_x^2. The kernel gives greater weight to gradients near the center of the 5 x 5 window.

Image

10	10	10	10	10	10
10	10	5	5	5	10
10	10	5	5	5	10
10	10	5	5	5	10
10	10	10	10	10	10
10	10	10	10	10	10

Kernel

1/12					
	-1	8	0	-8	1

$$I_x(2,2) = (-1 * 10 + 8 * 10 + 0 * 5 + -8 * 5 + 1 * 5) / 12$$

Figure 4: A sample gradient calculation for I_x.

Image

10	10	10	10	10	10
10	10	5	5	5	10
10	10	5	5	5	10
10	10	5	5	5	10
10	10	10	10	10	10
10	10	10	10	10	10

Kernel

1/12					
	-1	8	0	-8	1

$$I_y(2,2) = (-1 * 10 + 8 * 5 + 0 * 5 + -8 * 5 + 1 * 10) / 12$$

Figure 5: A sample gradient calculation for I_y.

Image

10	10	10	10	10	10
10	10	10	5	5	10
10	10	10	5	5	10
10	10	5	5	5	10
10	10	5	10	10	10
10	10	5	10	10	10

Current frame

Kernel

1/12					
	-1	8	0	-8	1

$$I_t(2,2) = (-1 * 5 + 8 * 5 + 0 * 5 + -8 * 5 + 1 * 10) / 12$$

Figure 6: A sample gradient calculation for I_t.

I_x^2

15	17	18	18	17	15
16	67	67	67	67	15
15	68	6	7	68	15
16	68	5	6	68	15
17	67	68	68	67	15
15	15	15	15	15	15

Kernel

1/256	1/64	3/128	1/64	1/256
1/64	1/16	3/32	1/16	1/64
3/128	3/32	9/64	3/32	3/128
1/64	1/16	3/32	1/16	1/64
1/256	1/64	3/128	1/64	1/256

$$\Sigma\, wI_x^2(0,0) = 15 * 1/256 + 17 * 1/64 + 18 * 3/128 + \ldots$$

Figure 7: A sample gradient smoothing for I_x^2.

Solving the Linear System. A linear system is constructed, as shown in figure 1, to contain all gradient information. Next, the 2 x 2 matrix containing gradient information for x and y (Figure 1) can be solved by first creating the quadratic equation shown in steps 1-3 of figure 8. This equation can then be solved for its two eigenvalues, λ_1 and λ_2, using the quadratic formula shown in figure 9. Once the eigenvalues are found, they are compared to a threshold, τ*=1,* and will fall into one of three cases:

Case 1: $\lambda_1 \geq \tau$ and $\lambda_2 \geq \tau$

- A is invertible and the system can be solved for the two velocity components, u and v using elementary linear algebra.

Case 2: $\lambda_1 \geq \tau$ and $\lambda_2 < \tau$

- A is singular and therefore non-invertible; the gradient flow is then normalized to give u and v.

Case 3: $\lambda_1 < \tau$ and $\lambda_2 < \tau$

- No optical flow can be computed; u and v are zero.

EVALUATION OF FLOW ACCURACY

The algorithm's implementation is evaluated based on its accuracy in computing optical flow. This evaluation looked at two main aspects: is the computed optical flow equivalent to the actual displacement of pixels between video frames and does the accuracy of the computed flow differ as pixel displacements increase between frames.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \Sigma\, wI_x^2 & \Sigma\, wI_xI_y \\ \Sigma\, wI_xI_y & \Sigma\, wI_y^2 \end{bmatrix}$$

1. $\det (A - \lambda I) = 0$

2. $(a_{11} - \lambda)(a_{22} - \lambda) - (a_{12})^2 = 0$

3. $\lambda^2 - \lambda(a_{11} + a_{22}) + a_{11}a_{22} - a_{12}^2$

Figure 8: The steps taken to create a quadratic equation for the 2 x 2 matrix A.

$$\lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$a = 1$

$b = -(a_{11} + a_{22})$

$c = a_{11}a_{22} - a_{12}^2$

Figure 9: The quadratic formula used to find two eigenvalues for the 2x2 matrix.

Materials

A set of 26 accuracy-test video sequences was created to test the accuracy of the algorithm. The test set contains a series of four videos, 176 x 144 pixels in size (QCIF videoconferencing size), containing a square of 40 x 40 pixels, which is composed of random colours. Each of these four videos shows the square moving at a rate of 1 pixel/frame either left, right, up, or down on a background composed also of random colours. These videos are 50 frames in length and play at a rate of 10 fps. Figure 10 shows the video of the square moving down; videos of the square moving left, right, or up are very similar. The accuracy-test set also includes 19 video sequences containing the same square moving left, except at speeds ranging from 2 pixels/frame to 20 pixels/frame.

Three of the accuracy-test video sequences were captured using a Winnov Videum PC Camera (Figure 10). The first contains an actor walking across the room from left to right. The second video contains an actor sitting in front of a computer working and moving his head left and right. The final video shows the same actor working while moving his head left and right, and now contains a second actor walking behind him across the scene from left to right. All three videos are 176 x 144 pixels in size, 75 frames in length, and play at a rate of 15 fps. Pixel displacements within each scene are unknown, and unfortunately each of these videos suffers from the capture effects of the Winnov camera.

a) Square moving down. b) Walking left to right.

c) Moving head left and right. d) Moving head and walking.

Figure 10: Four input videos (not shown at actual size).

Method

First, a series of four square test patterns was used to test the accuracy of the optical flow in each major direction: left, right, up, and down. Optical flow for each frame is stored in a file and also used as input for an output video sequence that visually displays the optical flow. The output video sequence is of the same frame size, but contains two frames less than the original video because for the last two frames a five-frame window does not exist and the algorithm is unable to compute flow. In the output video, red is used to indicate flow in *x* where a value of 255 indicates a magnitude of flow greater than or equal to 3, 170 indicates a magnitude of flow equal to 2, 70 indicates a magnitude of flow equal to 1, and 0 indicates no flow. The output video uses blue to show flow in *y* and uses the same value system as red. Each pixel's green component is set to zero. The flow stored in the output file is compared to the actual pixel displacements, and the output frame sequence is used to visually detect problems in computation.

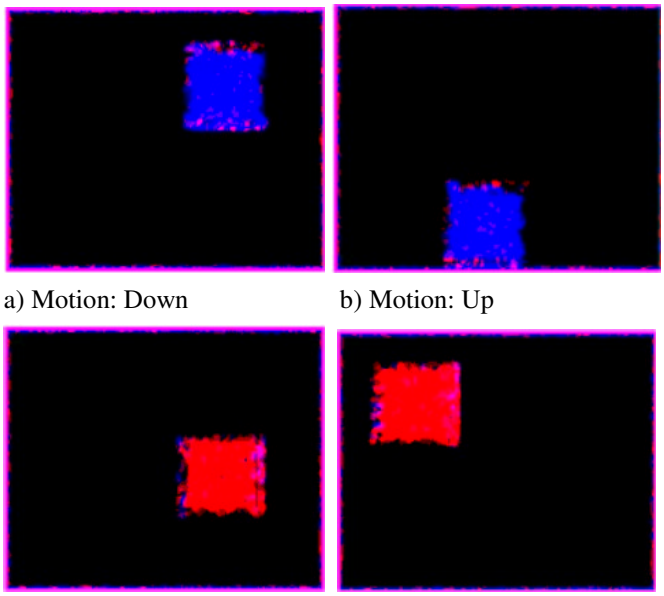
Next, 19 square test patterns were used to test the accuracy of flow at pixel displacements ranging from 2 pixels/frame to 20 pixels/frame. Optical flow for each frame is also stored in a file and the same output video sequences are created. Again, the flow stored in the output file is compared to the actual pixel displacements, and the output frame sequence is used to visually detect problems in computation.

Finally, 3 test patterns containing human actors are used to further analyze the accuracy of the output. Optical flow for each frame is again stored in a file and the same

output video sequences are created. This output video is used to visually analyze the accuracy of the flow, but flow values found in the file are not analyzed because the actual pixel displacements in the video are unknown.

Results

The algorithm is able to compute optical flow for all four video sequences containing the square moving at 1 pixel/frame with a large degree of accuracy. Flow for the moving pixels is computed within a 2-pixel range, i.e. for pixels moving at a rate of 1 pixel/frame, the algorithm either computed 1, 2, or 3 for the pixel's flow. An exception occurs for regions along the edge of the square where the algorithm is much less accurate. The output videos, shown in figure 11, display flow in *x* with red and flow in *y* with blue. Visually it is apparent that the flow is quite accurate and the algorithm is able to detect the correct size and shape of the moving object, as well as differentiate between flow in *x* and *y*.

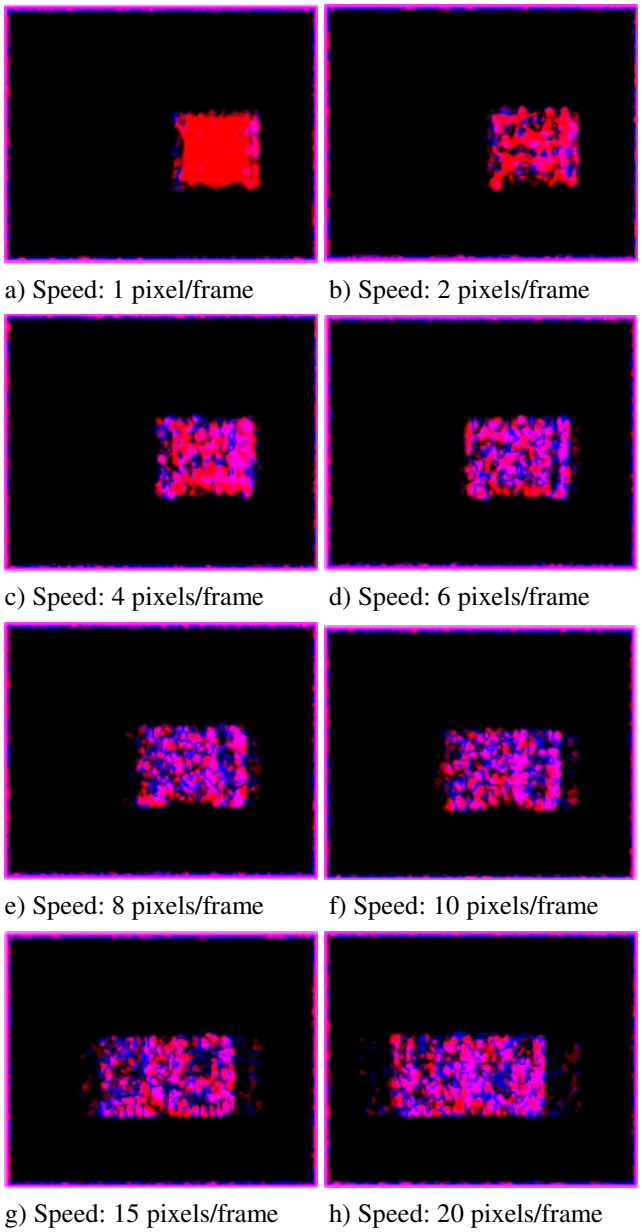


a) Motion: Down b) Motion: Up

c) Motion: Left d) Motion: Right

Figure 11: Four output videos of optical flow for a moving square (not shown at actual size).

The 19 test sequences that compare pixel displacements per frame show less accuracy in three respects. First, when pixel speeds are greater than 1 pixel/frame, the algorithm begins to fail in correctly differentiating flow between *x* and *y*. For the square moving left, flow artifacts begin to appear in *y* and this trend increases as speed increases. Figure 12 visually confirms this trend in optical flow output. An analysis of program execution shows that the majority of regions containing *y* artifacts also contain large eigenvalues and *u* and *v* are computed using the least squares method presented in the first eigenvalue case. Secondly, when pixel speeds are greater than 3 pixels/frame, the algorithm begins to fail in correctly identifying the shape of the moving object; flow artifacts begin to appear behind the moving square where no flow should be present. Again,



a) Speed: 1 pixel/frame b) Speed: 2 pixels/frame

c) Speed: 4 pixels/frame d) Speed: 6 pixels/frame

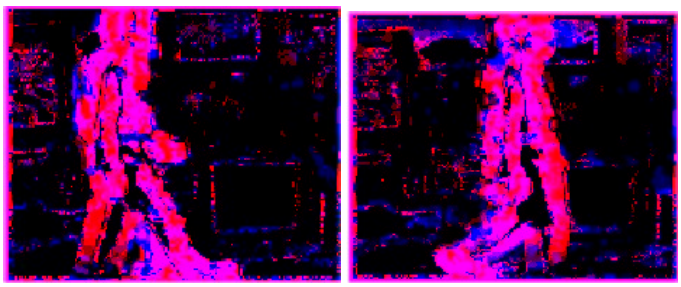
e) Speed: 8 pixels/frame f) Speed: 10 pixels/frame

g) Speed: 15 pixels/frame h) Speed: 20 pixels/frame

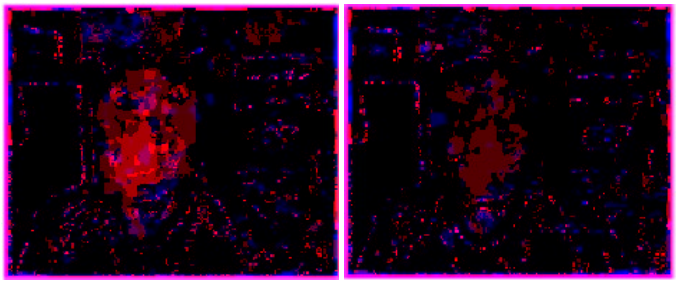
Figure 12: Eight output videos of optical flow for a moving square (not shown at actual size).

program execution shows that these regions also contain large eigenvalues and use the least squares method. Finally, when pixel speeds are greater than 4 pixels/frame the accuracy of the flow's magnitude begins to deteriorate and the algorithm computes the flow within a 4-pixel range, i.e. for pixels moving at a rate of 1 pixel/frame, the algorithm computes a flow magnitude between 1 and 5.

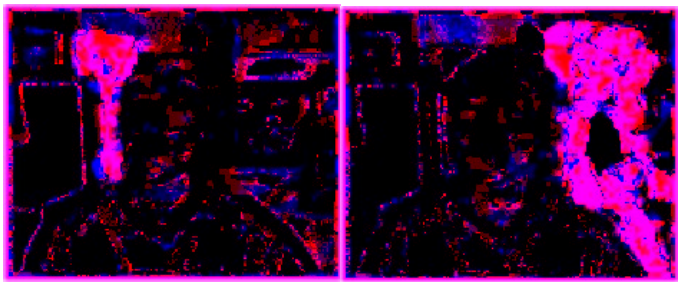
The final three test sequences contain unknown pixel displacements and the output flow values cannot be compared directly. Figure 13 shows the output flow video for each of the these test sequences. Visual analysis of the first video containing the actor walking shows that the majority of the movement is in *x*, which is correct. It also shows motion in *y* as the person walks, which accounts for slight movements in *y* during each stride. The second output video of the actor working while moving his head,



a) Walking left to right.



b) Moving head left and right.



c) Moving head and walking.

Figure 13: Three output videos of optical flow for moving actors, each shown at 2 points in time (not shown at actual size).

visually confirms that motion is almost solely in *x* and it is also apparent that these motions are very subtle as indicated by the light shades of red. The third video shows that the algorithm is capable of differentiating two different rates of motion: the faster motion of the walking person and the subtle head movements of the working person.

EVALUATION OF COMPUTATION PERFORMANCE

The algorithm's implementation is also evaluated based on its computational performance in computing optical flow. This evaluation compared the number of frames computable per second for varying frame sizes and monitored the processor's utilization.

Materials

A set of 7 computation-test video sequences was created to test the computation rate of the algorithm. Each video in the set contains a 40 x 40 square composed of random colours. The square moves at a rate of 1 pixel/frame over a background composed of random colours. Videos are 50 frames in length and play at a rate of 10 fps. The frame sizes selected for testing are:

1. 100 x 100
2. 160 x 120
3. 176 x 144 (QCIF)
4. 320 x 240
5. 352 x 288 (CIF)
6. 640 x 480
7. 720 x 480 (DV)

Method

Each of the 7 test videos is used as input to the optical flow algorithm. The total computation time for each frame size is recorded along with the corresponding frame per second processing rate. Each test video is processed for optical flow on a Dell workstation with a 1.4 GHz Intel Xeon processor and 512 MB of RAM running Windows XP as the operating system. Several background applications were running during testing, similar to typical scenarios when videoconferencing software is used.

Results

Processing rates range from ~20 fps for 100 x 100 size frames to ~0.5 fps for 720 x 480 size frames. Processing rates for all frame sizes are shown in figure 14. Ideal videoconferencing frame sizes of CIF and 320 x 240 were computed at a rate of ~1.8 fps and ~2.5 fps, respectively. During computation, the processor utilization was 100% for all frame sizes, as compared to ~10-30% utilization prior to computation.

DISCUSSION

In order to apply optical flow to videoconferencing systems that address privacy concerns, the optical flow algorithm must meet two main criteria. First, it should be accurate enough to distinguish regions of differing activity levels, for example, distinguishing regions containing someone working from regions containing someone walking into a room in the background. Second, it must be able to compute optical flow in real-time so that frames can be processed prior to being broadcast to others.

The optical flow implementation presented is capable of differentiating regions of different activity level. For example, figure 13c shows that the background motion of the person walking is much larger than the motion of the person who is working. Closer analysis of the computed flow shows that as the speed of objects increase, the accuracy of the algorithm deteriorates by first failing to completely distinguish flow in *x* from flow in *y* and then by failing to identifying the correct shape and size of the moving object. The algorithm was tested with speeds of up to 20 pixels/frame, yet it is unlikely that video containing people will have movement of this magnitude, especially if the intended use is capturing work activity. The algorithm fails to compute high displacement rates correctly because pixels begin to move outside the 5 x 5 window that is used to compute and smooth gradient estimations. If it were desirable to handle high displacements, a larger window size could be evaluated. For privacy applications, the flow produced is accurate enough.

The implementation is also seen to tax the workstation's processor quite heavily, yet it is clear that the

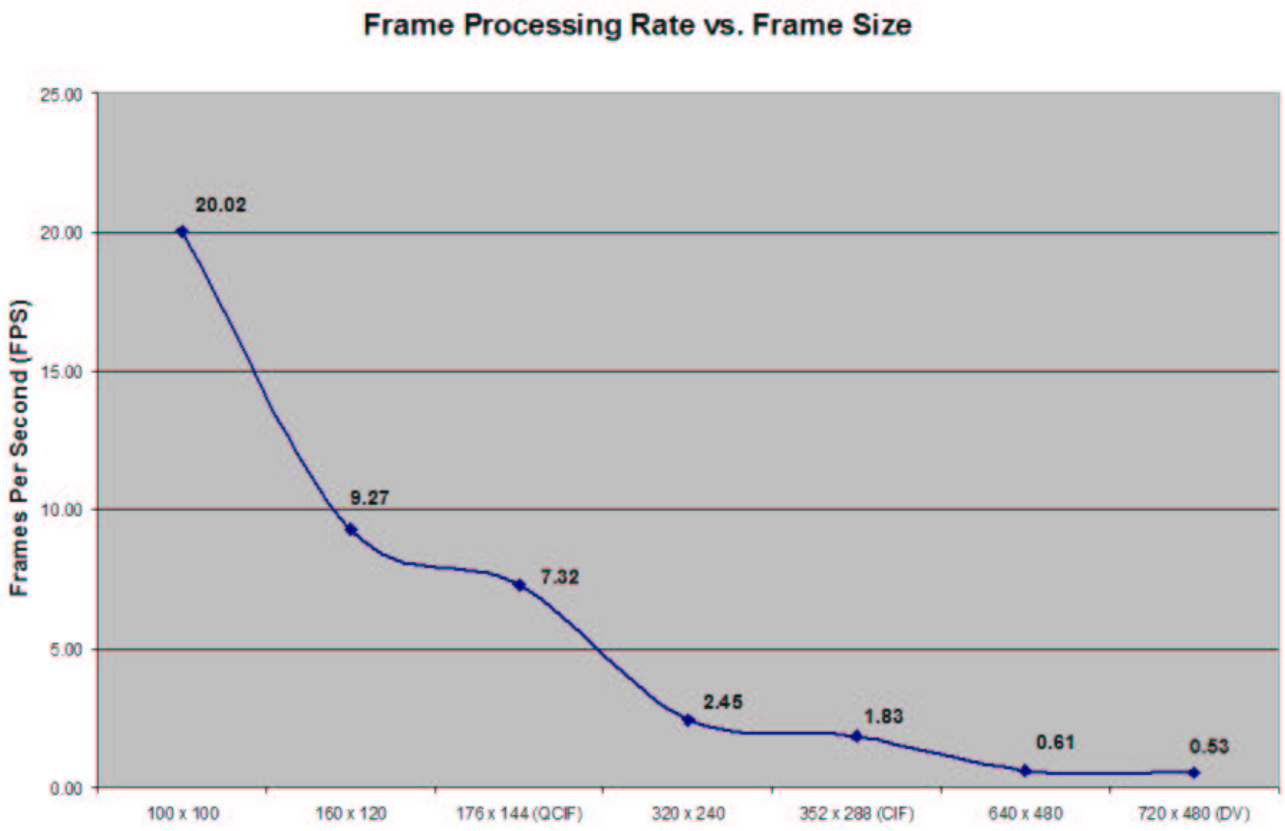


Figure 14: The processing rate of each frame size tested.

computational power of processors and other hardware peripherals will only increase in the future, thus easily remedying this problem. When using videoconferencing over the Internet, users are normally capable of obtaining frame rates between 5 and 10 fps, depending greatly on the Internet connection used and network congestion. As expected, when frame size increases so does the computation time and the corresponding frame per second flow output suffers as a result. The computation of QCIF frames and smaller frame sizes either fell within the ideal videoconferencing frame rates or exceeded its performance requirements. It is desirable, however, to use a frame size of 320 x 240 for videoconferencing, which is larger than QCIF frames. This size fell outside the desired rate, but it is expected that implementation optimization could improve its performance to a desirable level. Possible alterations may include optimizing calculations for specific processor technology and using less numerical precision during computation.

Although the flow output has not been used explicitly in any privacy preservation techniques, video frames that visually display flow present an interesting situation (Figures 11, 12, and 13). Flow frames naturally mask both the background details of a scene and the person's appearance in a scene. As well, a person's activity becomes partially masked – it is possible to understand the general activity of the person, but specific details are unknown (Figure 13). Flow output appears to offer its own form of privacy preservation, yet this idea still requires its own evaluation.

FUTURE WORK

The current implementation is capable of extracting frames from an AVI file and producing a second AVI that visually represents the optical flow computed. Future versions should be capable of using frames directly from an attached PC camera, especially if optical flow is to be applied as a tool for videoconferencing.

The current implementation shows that optical flow is capable of differentiating regions of a video containing varying levels of activity. Future implementations should use the calculated optical flow as input to a privacy preservation technique such as blur filtration, where regions of high activity could be blurred more than regions of low activity. Such a design could also be evaluated for its computational performance, as well as its effectiveness in preserving the privacy of multiple individuals.

CONCLUSION

Video is used by distance-separated workers, whether working from home or an office, for providing awareness information to others. This awareness is used to decide if and when to move into conversation or collaboration. When using video, however, information that may be considered threatening to one's privacy is broadcast to others. Optical flow presents a method for detecting varying degrees of activity in video, which could be used in conjunction with privacy preservation techniques. Such

techniques attempt to mask sensitive details in video where multiple people may be present. To be used with privacy sensitive videoconferencing systems, optical flow must be computable in a real-time and present a reasonable level of accuracy.

This paper presented an implementation of Lucas and Kanade's [13] differentiation method for computing optical flow in video. The implementation has been shown to compute flow in real-time for small image sizes and presents a desirable level of flow accuracy, capable of distinguishing regions varying in activity level. The algorithm does consume a large amount of system resources, however it is predicted that continuing technological improvements in computer hardware will resolve this problem. The implementation has yet to be used in conjunction with privacy preservation techniques, although it presents its own technique where videos visualizing the flow output naturally mask details of a person's appearance and the appearance of the captured location.

Acknowledgements. Thanks to Michael Boyle for his invaluable help and for the use of his collaborative multimedia toolkit.

REFERENCES

1. Barron, J.L., Fleet, D.J., Beauchemin. (1994), Performance of Optical Flow Techniques**.** *Proc. of International Joint Conference of Computer Vision,* Vol. 12, No. 1, pp. 43-77
2. Bellotti, V. (1996), What you don't know can hurt you: Privacy in Collaborative Computing**.** *Proc. HCI '96,* Springer, pp. 241-261.
3. Bellotti, V., and Sellen, A. (1993), Design for Privacy in Ubiquitous Computing Environments, in *Proceedings of the Third European Conference on Computer-Supported Cooperative Work (ECSCW'93)*, Kluwer Academic Publishers, Milan, pp. 77-92.
4. Bly, S., Harrison, S. and Irvin, S. (1993) Media spaces: Bringing people together in a video, audio, and computing environment. *Communications of the ACM 36(1),* ACM Press, pp. 28-46.
5. Boyle, M., Edwards, C. and Greenberg, S. (2000), The Effects of Filtered Video on Awareness and Privacy**,** *Proceedings of the CSCW'00 Conference on Computer Supported Cooperative Work* [CHI Letters 2(3)], ACM Press.
6. Fish, R., Kraut, R., Root, R., & Rice, R. (1992), Evaluating video as a technology for informal communication**,** *Proc. of CHI '92 Human Factors in Computing Systems,* New York: ACM Press, pp. 37-48.
7. Greenberg, Saul. (1996), Peepholes: Low Cost Awareness of One's Community*, ACM SIGCHI'96 Conference on Human Factors in Computing System, Companion Proceedings,* pp. 206-207.
8. Greenberg, S. and Kuzuoka, H. (2000). Using Digital but Physical Surrogates to Mediate Awareness, Communication and Privacy in Media Spaces*. Personal Technologies,* **4**(1)*,* January.
9. Hudson, S.E., and Smith, I. (1996), Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems, in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'96)*, Cambridge, MA.
10. Kraut, R., Egido, C., and Galegher, J. (1988), Patterns of contact and communication in scientific observation. *Proc ACM CSCW '88,* pp. 1-12.
11. Lee, A., Girgensohn, A., Schlueter, K. (1997) NYNEX Portholes: Initial User Reactions and Redesign Implications, *Group '97,* ACM Press, pp. 385-394.
12. Lim, S., Gamal, A., (2001) Optical Flow Estimation using High Frame Rate Sequences**,** *Proceedings of the 2001 International Conference on Image Processing*, v.2, p.925-928.
13. Lucas, B., and Kanade, T. (1981) An Iterative Image Registration Technique with an Application to Stereo Vision**,** *Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI),* pp. 674-679.
14. Mantei, M., Baecker, R., Sellen, A., Buxton, W., Milligan, T., and Wellman, B. (1991) Experiences in the use of a media space**.** *Proc. of CHI '91 Human Factors in Computing Systems,* New York: ACM Press, pp. 203-209.
15. Tang, J.C., Isaacs, E., and Rua, M. (1994). Supporting Distributed Groups with a Montage of Lightweight Interactions. *Proc. of the ACM Conference on Computer-Supported Cooperative Work, CSCW '94*, ACM Press, pp. 23-34.
16. Whittaker, S., and O'Conaill, B. (1997), The Role of Vision in Face-to-Face and Mediated Communication, in *Video Mediated Communication*, Finn, Sellen, and Wilbur eds., LEA Press.
17. Zhao, Q.A., and Stasko, J.T. (1998), Evaluating Image Filtering Based Techniques in Media Space Applications**,** in *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'98)*, Seattle, pp. 11-18.