Geometric Sequence (GS) Imaging with Bayesian Smoothing for Optical and Capacitive Imaging Sensors

Kuntal Sengupta
MERL
201 Broadway, Cambridge, MA 02139

Fatih Porikli
MERL
201 Broadway, Cambridge, MA 02139

Abstract

In this paper, we introduce a novel technique called Geometric Sequence (GS) imaging, specifically for the purpose of low power and light weight tracking in human computer interface design. The imaging sensor is programmed to capture the scene with a train of packets, where each packet constitutes a few images. The delay or the baseline associated with consecutive image pairs in a packet follows a fixed ratio, as in a geometric sequence*. The image pair with shorter baseline or delay captures fast motion, while the image pair with larger baseline or delay captures slow motion. Given an image packet, the motion confidence maps computed from the slow and the fast image pairs are fused into a single map. Next, we use a Bayesian update scheme to compute the motion hypotheses probability map, given the information of prior packets. We estimate the motion from this probability map. The GS imaging system reliably tracks slow movements as well as fast movements, a feature that is important in realizing applications such as a touchpad type system. Compared to continuous imaging with short delay between consecutive pairs, the GS imaging technique enjoys several advantages. The overall power consumption and the CPU load are significantly low. We present results in the domain of optical camera based human computer interface (HCI) applications, as well as for capacitive fingerprint imaging sensor based touch pad systems.*

1. Introduction

Motion estimation from an image sequence is one of the primary components in a tracking system, specifically designed for vision based human computer interfaces (HCI). The imaging sensor, optical or otherwise, is now the input to the HCI system for the purpose of basic mouse/touchpad type control. Throughout this paper, we will refer to such mouse/touchpad control tasks as *navigation* tasks. In navigation tasks, it is important to have an accurate estimation of the motion, starting from the images captured by the sensor. For applications such as the fine placement of a cursor in the screen, accurate estimation of small motion is very important. On the other hand, for moving quickly from one part of the screen to the other (specifically in gaming applications), estimation of fast motion is crucial in realizing a pleasant user experience.

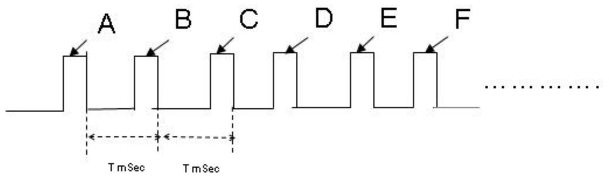


Figure 1. The timing diagram here illustrates a train of images that samples the scene densely in time.

A straight forward solution to realizing a system that can reliably capture both fast and slow motion is to image continuously at a very high frequency. By doing this, we are always in a position to estimate the fast motion. To compute the parameters of the slow motion accurately, one needs to choose image pairs significantly far away in time. This is illustrated by the timing diagram in Fig. 1. In this example, images A and B can be used to compute the fast motion. However, if the object is moving slow, image pairs A and D could probably be an ideal candidate of choice.

We list the problem of imaging continuously at high frequency below:

- **Large power consumption**: Imaging sensors (for example capacitive fingerprint sensors) that have a high ratio between ON and IDLE current consumption will drain excessive power. This is an important factor to consider when such sensors are integrated in mobile devices, such as the cell phones and laptops.
- **Large data transfer rate**: Typically, if the HCI application is running in an embedded environment, continuous imaging as in Fig. 1 would imply transferring image data to the host processor every T ms.

- **Large CPU load**: Typically, in gaming applications, the CPU is primarily busy servicing computationally expensive computer graphics algorithms, and the CPU load budgeted for simple navigation tasks is usually very minimal (typically less than 5 % of the total CPU load). Imaging every T ms would imply a larger number of image pairs to correlate. Hence, a large CPU load needs to be budgeted for the purpose of motion estimation.

In many HCI tasks, the velocity of the object of interest is relatively constant, or changes slowly. For this, it may suffice to sense images as shown in Fig. 2(a). Here two images are captured with a delay of T between them, and the pattern is repeated over time. Hence motion measurements can be made every T_L mSec, where $T_L > T$.

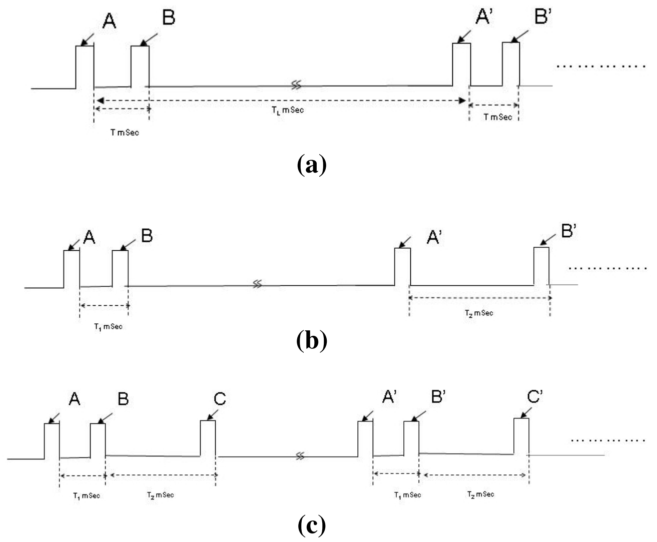


Figure 2. Three different ways to sparsely sample the scene in time.

For very small values of T, the system is ideal for capturing fast motion. When T is relatively large, we have a system that measures slow motion. However, it cannot measure fast motion reliably. Other alternate ideas could be a mix and match of the techniques in Fig. 1 and Fig. 2(a), where we alternate between fast and slow measurements. This is illustrated in Fig. 2(b). Alternatively, as in Fig. 2(c), we could use motion computed between pairs A and B for fast motion detection and A and C for slow motion computation. The timing diagram in Fig. 2(c) forms the basis of our geometric sequence (GS) imaging technique that we will present in the next section. The group of images, (A, B, C), is formally called as a *packet*. For $T_L = T_2$, the example in Fig. 2(a) is similar to the example in Fig. 2(c), where the frame A' can double up as frame C.

An idea similar to the GS system appears in the patent disclosure in [1]. The system described in [1] is functional in millions of biometric cell phones, where the fingerprint sensor doubles as a touchpad. The paper discussed here is a significant extension of the GS type system discussed in [1]. The paper here introduces a Bayesian-GS imaging theory. We need to bear in mind the issues of repeated scene texture and other non idealities, such as noise and illumination variation, that can lead to erroneous estimation of velocity. The theory of motion estimation present in [1] is mainly ad hoc, and not grounded on strong principles of Bayesian tracking as presented in this paper. To ensure a smooth estimate of the velocity profile, we present a method to propagate the motion likelihood over time. For this, we generate the confidence maps computed from the pairs $(A,\ B)$ and $(B,\ C)$, and combine them to generate an overall motion confidence map for the packet. We discuss this in Section 3. Next, using a Bayesian approach, we estimate the smooth motion. This is discussed in Section 4. This is followed by the experimental results and the conclusions section.

1.1. Prior work in motion tracking for HCI

Motion estimation for HCI is an actively researched area. The paper by Moeslund *et al* [2] describes a comprehensive survey of computer vision-based human motion capture literature from the past two decades. The authors provide a taxonomy of system functionalities, broken down into initialization, tracking, pose estimation, and recognition. A similar survey by Gavrila on the analysis and interpretation of human motion appears in [3]. The system described by Turk in [4] and the Pfinder system by Wren *et al.* [5] followed by the Spfinder system is perhaps one of the earlier complete human body tracking system, used for animating computer generated characters. As for hand based HCI systems, Palovic *et al.* [6] provide a comprehensive survey of hand gesture based input systems. There is a plethora of vision based hand/finger tracking systems in the literature. Wu *et al.* [7] present a method for tracking the 3D position of a finger, employing a single camera placed several meters away from the user. Segen and Kumar describe the GestureVR system in [8] that uses a stereo pair of calibrated cameras for tracking hand gesture. Rehg and Kanade in [9] describe DigitEyes, a model-based hand tracking system. Unfortunately, none of these works address the issue of low computational load and low power solutions, suitable for embedded devices, such as mobile phones. Processing of every frame is assumed in these HCI systems.

Dealing with multi camera multi frame systems in a methodical way has however appeared in stereo vision and computational photography literature. In [10], the authors describe how multiple baseline stereo information can be fused to deal with problems inherent with scenes with repeated patterns or structures. Raskar *et al* [11] present a coded exposure approach to combine information in multiple images captured with variable exposure time to remove the blur in an image. Although not relevant to the HCI motion estimation problem, such concepts of fusing images taken at different (known) time instances has inspired our work in this paper.

1.2. Domain of application

The basic theory of GS imaging presented here generalizes to most motion tracking problems. However, we would now define the scope of the work presented here. Human computer interface and gaming applications are growing in mobile devices such as cell phones and PDAs. Moving the cursor, scrolling the menu, and navigating in a gaming application are some of the application areas that we are addressing here.

As shown in Fig. 3(a), the camera of the cell phone is directed towards a relatively planar region in the scene. The motion of the cell phone with respect to the scene acts as an input to the device. For simplification purposes, we will assume pure translation motion in this paper. However, the theory can be extended for more generalized motion models to handle rotations, and arbitrary scene geometry.

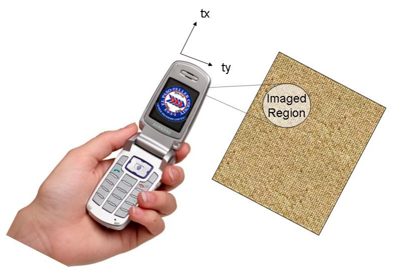
In Fig. 3(b), we illustrate yet another application of motion tracking in navigation tasks. Here, the finger is moved on a 500 dpi capacitive fingerprint sensor, as in a touchpad. A small 32×32 section of the fingerprint sensor is activated in the *video mode* to image the ridge valley pattern. As shown in the picture, at any time instance, the conductive layer of the skin acts as one of the plates of the capacitors. The equipotential lines in this thin plate arrangement follows the pattern of the ridges and the valleys. The pixel sensor plate array picks up the ridge/valley pattern of the equipotential lines, leading to the highly textured images of a small region of the finger. The ridges and valley patterns in the finger provide sufficient texture variation to perform reliable motion computation and tracking, as the finger moves on the sensor.

In this paper, the theory of GS imaging is limited to the following assumptions.

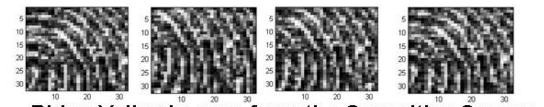
- There is only one global motion that the scene undergoes relative to the camera.
- The motion is purely translational in the x and the y direction.

2. Geometric Sequence (GS) Imaging: The Basics

To address the light weight (low CPU load), low power requirements of the navigation system for HCI, we have designed a novel geometric sequence (GS) imaging technique.



(a)



(b)

Figure 3. Illustration of two typical scenarios where motion tracking is required for HCI tasks. In (a), the cell phone is moved with respect to the world. In (b), the finger is moved on a small capacitive fingerprint sensor and we gather a sequence of small 32x32 frames containing ridge/valley patterns.

The GS technique is applicable for optical cameras as well as for other imaging sensors, like capacitive fingerprint sensors. In this technique, instead of imaging every T ms, we perform packet imaging, as shown in Fig. 2(c). Here, we collect an image packet every T_L ms. Inside the packet, we have two image frames with a short time delay T_1 between them , and two frame pairs imaged with long time delay T_2 between them. Note that $T_2 > T_1$. Fig 2(c) shows three frames in the packet; A, B, and C, respectively. In principle, we are not limited to imaging only three images. It is possible to image m images in general, where the gap between the kth and the $(k+1)$th image is T_k, and

$$\frac{T_{k+1}}{T_k} = r. \quad (1)$$

We estimate the fast and the slow motion by correlating the frame pairs $(A,\ B)$ and $(B,\ C)$, respectively. We refer to the pair $(A,\ B)$ as the *fast pair* and the pair $(B,\ C)$ as the *slow pair*.

Next, we introduce the parameters in the imaging system that would help us in quantifying the performance of the motion detection scheme. Let v_x and v_y be the x and the y components of the velocity of the object in terms of the image coordinate system. The units of v_x and v_y are in pixels per unit time.

The parameters d_{xf} and d_{yf} are the maximum absolute values of the x and y displacement that we choose to estimate. In other words, given two frames A and B, we wish to

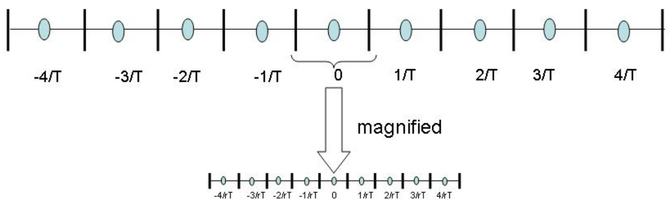


Figure 4. An illustration of the velocity range captured by the fast pair (top) and the slow pair (bottom).

search over all displacement hypothesis in the range shown below

$$\left|\hat{m_{x f}}\right| \in \left[-d_{x f},\ -(d_{x f}-1), \ldots,\ -1,\ 0,\ 1,\ 2, \ldots,\ d_{x f}\right],$$

and

$$\left|\hat{m_{y f}}\right| \in \left[-d_{y f},\ -(d_{y f}-1), \ldots,\ -1,\ 0,\ 1,\ 2, \ldots,\ d_{y f}\right].$$

Here $\hat{m_{x f}}$ and $\hat{m_{y f}}$ are the estimated x and y components of the displacement of the object in the fast frame pair. For the slow pair, the parameters $d_{x s}$ and d_{y_s} correspond to the the maximum absolute displacement values estimated in the x and the y direction, respectively. Also, $\hat{m_{x s}}$ and $\hat{m_{y_s}}$ are the estimates of the displacement for the slow pair.

Assuming a constant velocity in the image packet, we observe that

$$\hat{m_{x f}} = r\hat{m_{x s}} \qquad \hat{m_{y f}} = r\hat{m_{y_s}}.$$

We now introduce the notion of the *dead zones*. Dead zones are the range of motion values that cannot be measured by the designed system. For this, we refer to Figure 4. In this figure, we illustrate the range of motion values in 1D sensed by the fast pairs. Note that for each value of $m_{x f}$, we have a range of velocities that map into that range. For example, $m_{x f}=0$ represents the velocity in the range $[\frac{-0.5}{T}\ \frac{0.5}{T}]$. Similarly, for an arbitrary value of motion $m_{x f}=a$, we sense motion in the range $[\frac{a-0.5}{T}\ \frac{a+0.5}{T}]$. The maximum velocity in the x and the y direction that can be measured are

$$\max(v_x) = \frac{d_{x f}+0.5}{T} \quad \text{, and} \quad \max(v_y) = \frac{d_{y f}+0.5}{T}$$

respectively.

Similarly, for the slow pairs, the maximum velocity in the x and the y direction that can be measured are

$$\max(v_x) = \frac{d_{x s}+0.5}{rT} \quad \text{, and} \quad \max(v_y) = \frac{d_{y_s}+0.5}{rT}$$

respectively.

To estimate small motion precisely for fine positioning applications, we would be using the measurements from the slow pair, and for fast motion, we use the measurements from the fast pair. To ensure enough overlap between the two measurement *apparatus*, and a smooth transition between them, we need to have sufficient overlap in the range of velocities that each apparatus measures. For example, the rule we follow is that if we measure that $\hat{m_{x f}}=0$, we use the measurement $\hat{m_{x s}}$. The maximum velocity measured by the slow pair has to exceed the maximum velocity value corresponding to the measurement of 0 displacement in the fast pair. Thus, to avoid dead zones, or in other words, to have a continuous handover from the slow pairs to the fast pairs and the vice-versa,

$$\frac{d_{x s}+0.5}{rT} > \frac{0.5}{T}.$$

Simplifying the above equation,

$$d_{x s} \quad > \quad 0.5r - 0.5.$$

For $d_{x s}=4$, $d_{y f}=4$, and $r=4$, it is easy to verify that the above inequality is satisfied.

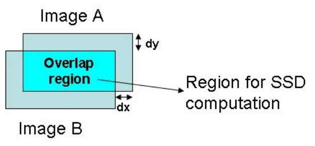


Figure 5. Sum of Squared Distance (SSD) is computed in the overlap region for image pairs A and B.

Given a frame pair, we minimize the sum of the squared distances (SSD) to compute the displacement estimates. The fast SSD table has a total of $(2d_{x f}+1)\times(2d_{y f}+1)$ entries. As shown in Fig. 5, for a given motion hypothesis between images A and B, we compute the sum of the squared distance between the overlap region, and normalize it. The displacement hypothesis $(\hat{m_{x f}},\ \hat{m_{y f}})$ with the minimal value is chosen as the velocity of the object. Alternatively, as in [12], the SSD values are related to the probabilities as follows. Thus

$$P\left(\hat{m_{x f}}=d_1,\ \hat{m_{y f}}=d_2\right) \propto e^{-\alpha SSD_f(d_1,d_2)}. \qquad (2)$$

For the slow pair $(B,\ C)$, we compute the SSD table SSD_s, which is of dimension $(2d_{x s}+1)\times(2d_{y_s}+1)$.

For each packet, we have two SSD tables, SSD_f and SSD_s, respectively. Next, we discuss a way of combining the SSD tables as a pre processing step to obtaining smooth estimates of the velocity parameters.

3. Combining the SSD Tables for Velocity Estimation

Given a slow and a fast table, the simplest way to obtain a combined decision is to observe the fast table SSD_f initially. If the entry corresponding to the motion $(0, 0)$ is a winner (i.e, has the least SSD value), then we observe the table SSD_s. The entry corresponding to the lowest SSD values in the slow table is declared as the winner. If this is $(m_{xs},\ m_{y_s})$, our estimates for the x and the y components of the velocity are

$$v_x = \frac{m_{xs}}{rT} \qquad \text{, and} \qquad v_y = \frac{m_{y_s}}{rT}. \tag{3}$$

In the event that the location $(0, 0)$ in the table SSD_f is not the winner, we record the winning location in this table, say (m_{xf}, m_{yf}), and $v_x = m_{xf}/T$ and $v_y = m_{yf}/T$.

However, errors in the SSD computation due to presence of noise and aliasing due to repeated structures could cause an erroneous estimate of the motion hypothesis. The effect of erroneous fast motion computation is most noticeable in navigation applications. To address this problem, we adopt a Bayesian smoothing approach. The approach requires an overall motion likelihood table to be computed for a given packet. Hence, we need to merge the two SSD tables to form one table.

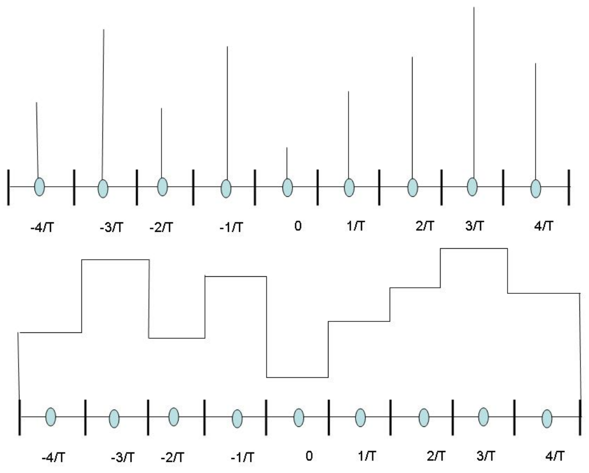


Figure 6. Interpolation of the SSD values for the fast table are done using the nearest (quantized) motion value for which image based SSD is computed.

As in the previous section, for ease of explanation, we would be reverting to the 1D case. In Fig. 6, we illustrate the SSD values interpolated for $v_x \in \left[-\frac{d_{xf}+0.5}{T}\ \frac{d_{xf}+0.5}{T}\right]$, corresponding to the fast table. In general, for 2D SSD table, $SSD(v_x, v_y) = SSD_f(round(\frac{v_x}{r}),\ round(\frac{v_y}{r}))$. This is the simplest extension of the SSD from a discrete set of velocity values to a continuous set of velocity values. Bilinear interpolation, as shown in Fig. 7 yields a continuous solution, and we adopted this for all our experiments. However, for the ease of illustration, we will continue with the interpolation as in Fig. 6.

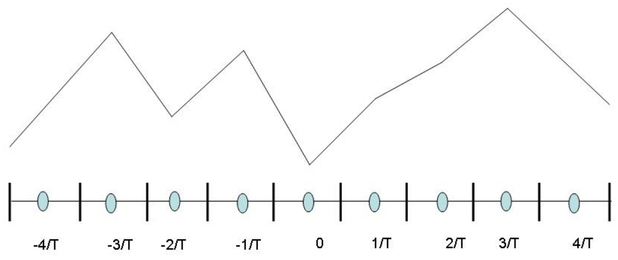


Figure 7. Bilinear interpolation of the SSD value.

Note that the slow SSD table gives sensible results only when the object motion is slow. In other words, only when $m_{xf} = 0$, the SSD_s table can be considered usable. Merely replacing the values of the SSDs from the slow table into their equivalent position in the fast tables would be incorrect. There is a need to *equalize* the two tables, before the slow table can be inserted into the fast table. For this, each of the SSD values in the slow table are multiplied by a factor β, where

$$\beta = \frac{SSD_f(0)}{\frac{1}{|S|}\sum_{m_x=\lceil -0.5r\rceil}^{m_x=\lceil 0.5r\rceil} SSD_s(m_x)}. \tag{4}$$

In the above equation, S corresponds to the set of motion displacements $\{-m, -(m-1), \ldots,\ (m-1),\ m\}$, where $m = \lceil 0.5r\rceil$. In other words, for the overlap region in the velocity space, the mean SSD of the slow table is equated to the mean SSD corresponding to the fast table. The fused SSD table after scaling the slow table is illustrated in Fig. 8. For a non zero value of m_{xf}, we use the interpolated version of the fast SSD table as shown in Fig. 6.

Next, the SSD values are converted to probability tables. For this, we use Eqn. 2, and rescale the probability table such that it sums to unity. We have experimented with several values of the α parameter.The choice of $\alpha = 0.01$ has been used found to be optimal, and has been used for our experiments.

4. Bayesian Smoothing in GS Imaging System

As mentioned earlier, just relying on the SSD tables to estimate the velocity parameters could lead to noisy estimates in presence of noise, and for various other reasons, such as presence of repeated structures in the scene.

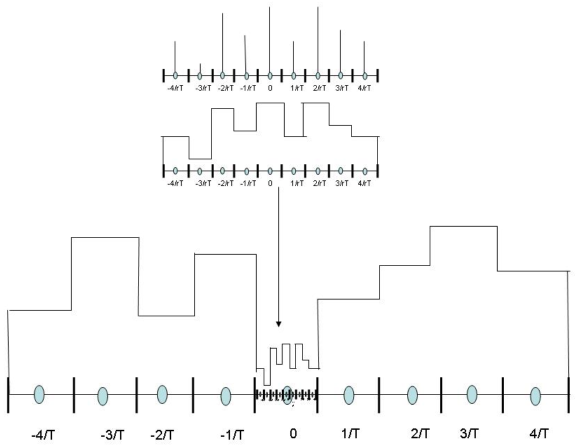


Figure 8. The overall (fused) SSD table when $m_{x\,f} = 0$.

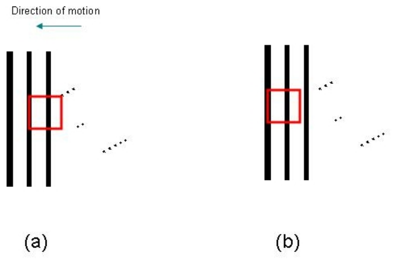


Figure 9. Example of a situation where the SSD computation between the two frames (shown in square) would lead to an inaccurate estimate of the motion.

We illustrate one such example in Fig. 9. Here, we observe that the object has repeated picket fence like patterns in the middle of the scene. When the sensor traverses from left to right, the random dot patterns in the scene leads to the correct estimation of the velocity. However, on entering the ridge like pattern, the motion estimation could easily be confused between a positive motion in the x direction versus a negative motion. If we preserved the previous states of the motion vector, it would be possible to obtain a smooth estimate.

Assume that we have already estimated the smooth velocity $v_x(t)$, where t is an index on the image packet. On arrival of the t th packet O_t, we compute the SSD table and the probability table as shown in Eqn. 2. Our Bayesian update and estimation technique is loosely based on prior Bayesian tracking techniques in the vision literature. One such example is the Condensation framework presented by Isard *et al.* in [13]. Using Bayes rule and assuming a first

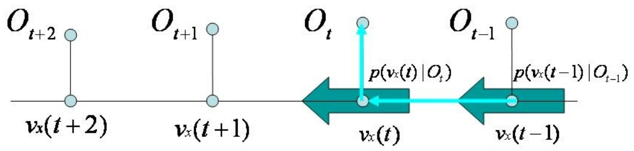


Figure 10. Here we illustrate the first order Markov Model for the random process $v_x(t)$.

order Markov model for the random process $v_x(t)$, we have

$$P\left(v_x(t) = v\,|O_t, O_{t-1}, \ldots, O_1\,\right) \propto P(O_t|v_x(t) = v)\times$$

$$\sum_{\forall v'} P(v_x(t) = v|v_x(t-1) = v')P\left(v_x\left(t-1\right) = v'\,|O_{t-1}, \ldots, O_1\,\right). \quad (5)$$

We illustrate the first order Markov process in Fig. 10. Note that $P(O_t|v_x(t) = v)$ is an entry in the probability table derived in Eqn. 2.

Assuming constant velocity model, and a Gaussian noise model

$$v_x(t) \;=\; v_x(t-1) + n(t).$$

where $n(t)$ is a Gaussian process, we observe that the term $\sum_{\forall v'} P(v_x(t) = v|v_x(t-1) = v')P\left(v_x\left(t-1\right) = v'\,|O_{t-1}, O_{t-2}, \ldots, O_1\,\right)$ in Eqn. 5 is a convolution of a Gaussian kernel with the table $P(v_x(t-1) = v|O_{t-1}, \ldots, O_1)$ that was estimated at time $t-1$. In our application, the Gaussian table is assumed to be a 3×3 matrix G, where

$$G = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$

The recursive algorithm for obtaining the probability table corresponding to the velocity hypotheses is as follows:

- Given a motion sensing packet at time t, compute the probability table $P(O(t)|v_(t) = v)$, for $v \in \left[-\frac{d_{x\,f}+0.5_f}{T}, \frac{d_{x\,f}+0.5_f}{T}\right]$ using the SSD fusion scheme and converting the SSD table into the probability map as in Eqn. 2.
- For $t = 0$, $P(v_x(t) = v|O(0)) = P(O(0)|v_x(0) = v)$.
- For $t > 0$, smooth the estimated probability table from the previous iteration, $P(v_x(t-1) = v|O_{t-1}, \ldots, O_0)$ by convolving it with G.
- Compute the state probability table for the current time instance by multiplying the table $P(O(t)|v_x(t) = v)$ obtained in step 1 with the smoothed table from step 3, as shown in Eqn. 5.

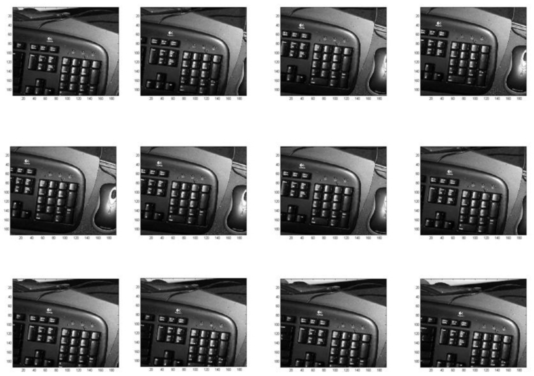


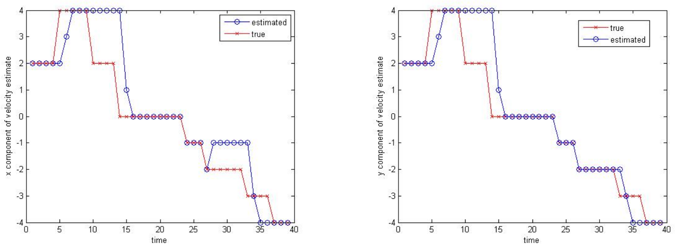
Figure 11. The synthetic image sequence generated.

The velocity estimate at time t is

$$v_x = \arg\max_v P(v_x(t) = v | O_t, O_{t-1}, \ldots, O_1).$$

5. Experimental Results

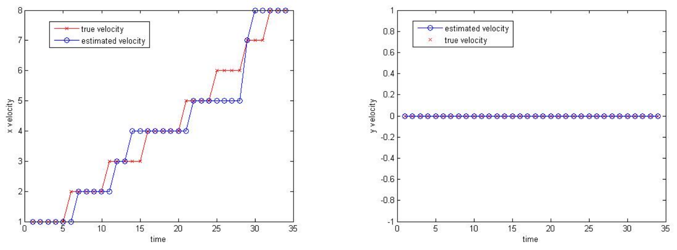
To test the effectiveness of the algorithm, we formulated a few controlled experiments. We captured a high resolution picture of the scene, and simulated image frames taken from a moving (virtual) camera. We did this by cropping smaller images from the high resolution image and adding noise to them, as the virtual camera moved. We generated motion sequences for known velocity values, and benchmarked the performance of the GS imaging with respect to the ground truth values. In this experiment, the ratio in GS imaging is $r = 4$. Also, $T = T_L$, implying that we took images of pattern A_1B_1, $A_2B_2, \ldots$; and used (A_i, B_i) as the slow pair, and (B_iA_{i+1}) as the fast pair. The delay between B_1 and A_2 is 4 times the delay between A_1 and B_1.



(a) (b)

Figure 12. The plot of the estimated velocity and ground truth velocity vs. time is shown. We plot the X and the Y components in (a), and in (b), respectively.

In Fig. 11, we show twelve representative images in the sequence collected. Here, the camera was initially moved diagonally, with slow motion, and gradually accelerating to large values, followed by a reversal in direction. Although our motion model in the previous section assumes constant velocity model plus noise, we keep up with small acceleration components because of the Gaussian smoothing of the probability table computed from previous packets. The estimated velocity components in the X and the Y direction are illustrated in Fig. 12. The ground truth velocity (in red x) is overlaid on the estimated velocity (in blue o). One can note that the estimated velocity closely tracks the ground truth velocity. One interesting thing to note is the jump of the velocity estimate from 2 to 4. Essentially, when the slow pairs perceives a motion of more than 2 pixels, the fast pair also perceives a non zero motion. In our SSD table fusion scheme, we then disregard the contribution of the slow table. Thus, the fast displacement of 1 unit (which is equal to 4 units in terms of the slow pair) emerges as the winner. In Fig. 13, we illustrate the results of velocity estimation in the case of pure horizontal motion. Here, the camera underwent slow as well as fast motion. Note that the system can estimate slow as well as fast motion reliably.

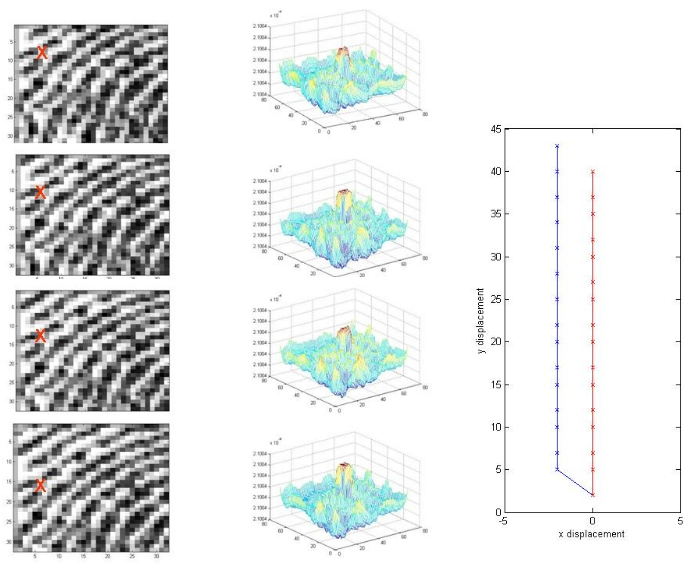


(a) (b)

Figure 13. The estimated/ground truth velocity components in the X and the Y direction are shown in (a) and (b), respectively for a sequence collected while the camera undergoes horizontal motion.

Next, we present results from the sequence collected from a capacitive fingerprint sensor. The images captured are of dimensions 32×32, and are of 500 dpi resolution. The fast pair delay is 1 mSec and the slow pair delay is 4 mSec. For 500 dpi images, this translates to a maximum speed capture of 20 cm/sec, for $d_{xf} = 4$. The slowest pair can capture motion fine motion as small in magnitude as 0.32 cm/sec. The packets are repeated every 30mSec, implying that we have a velocity estimate at approximately 30 Hz, which is frequent enough to move a cursor in a screen.

Figure 14(a) shows four sample image frames, one per packet, captured by the capacitive sensor as the finger was moved vertically upwards (a minutiae is pointed out with a red x). The fused probability maps computed after receiving a packet is shown in Figure 14(b). The estimated velocity using the Bayesian scheme, versus a simplistic (non Bayesian) scheme is shown in Fig. 14(c). In the simplistic scheme, the winner is selected from the fused SSD derived from the current packet. Note that propagating the probability map using the Bayesian framework helps us in obtaining a smooth estimate, and we avoid sudden jumps in the



(a) (b) (c)

Figure 14. (a) One image frame from each of the four representative packets are illustrated here, as the finger moves vertically upwards. (b) Here we show the motion hypotheses probability maps estimated after computing the fused SSD table from the observed packet, followed by the Bayesian update. (c)The estimate of the trajectory using the Bayesian update scheme is shown by the straight line to the right, and a simple scheme without the Bayesian update is shown by the line to the left.

velocity profile caused due to repeated ridge valley pattern (leading to aliasing).

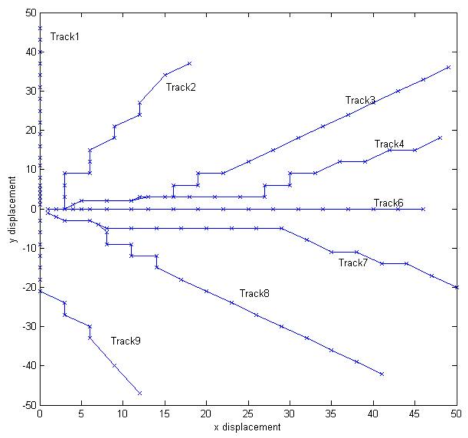


Figure 15. (The trajectory estimates as the finger is swiped in nine different directions.

In Fig. 15, we plot the tracks computed from the estimated finger motion under nine different situations. Track1 is an estimate of the track for a pure vertical motion, Track 2 is the estimate corresponding to the NorthNorthEast finger motion, and so on. The motion estimation using the GS scheme is indeed reliable enough to be used in navigational application in HCI.

6. Conclusions

In this paper, we present a novel Geometric Sequence Imaging technique, by imaging with a train of packets comprising image pairs to capture fast and slow motion. The technique has been applied on both optical as well as non optical imaging sensors. Future extensions of this work include imaging scenes with objects moving around at various speeds, and tracking each of the detected objects using the Bayesian tracking framework.

References

[1] J. A. Tykowski, J. W. Neil, and K. Sengupta, Finger Sensing Device for Navigation and related Methods, Patent Disclosure WO/2006/044815.

[2] T. B. Moeslund and Erik Granum, "A Survey of Computer Vision-Based Human Motion Capture," *Computer Vision and Image Understanding* , vol. 81, no. 3, pp. 231-268, 2001.

[3] D. Gavrila, "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding,*, vol. 73, no. 1, pp. 82-98, 1999.

[4]M. Turk, "Computer vision in the interface", *Commun. ACM*, vol. 47, no. 1, pp. 60-67, 2004.

[5] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *IEEE Trans. PAMI*, vol. 19, no. 7, pp. 780-785, 1997.

[6]V. Pavlovic, R.Sharma and T. S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. PAMI*, vol. 19, no. 7, pp. 677-695, 1997.

[7]A.Wu, M. Shah and N. da Vitoria Lobo, "A Virtual 3D Blackboard: 3D Finger Tracking Using a Single Camera,"', *Proc. FG*, pp. 536-543, 2000.

[8]J. Segen and S. Kumar, "Gesture VR: Vision-Based 3D Hand Interface for Spatial Interaction," *Proc. ACM Multimedia*, pp. 455-464, 1998.

[9]J. M. Rehg and T. Kanade, "Visual Tracking of High DOF Articulated Structures: an Application to Human Hand Tracking,"' *Proc. ECCV*, pp. 35-46, 1994.

[10]M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo," *IEEE Trans. PAMI*, vol. 15, no. 4, pp. 353-363, 1993.

[11]R. Raskar, A. K. Agrawal and Jack Tumblin, "'Coded exposure photography: motion deblurring using fluttered shutter," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 795-804, 2006.

[12]K. Nickels and S. Hutchinson, "Estimating uncertainty in SSD-based feature tracking," *Image and Vision Computing*, vol. 20, no. 1, pp. 47-58, 2002.

[13] M. Isard and A. Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5-28, 1998.