Representing local motion as a probability distribution matrix and object tracking

Yoav Rosenberg

Institute of Computer Science
The Hebrew University
Jerusalem Israel 91904
yoavr@cs.huji.ac.il

Michael Werman

Institute of Computer Science
The Hebrew University
Jerusalem Israel 91904
werman@cs.huji.ac.il
http://www.cs.huji.ac.il/~ werman/

Abstract

Due to the aperture problem, local motion between two images can be precisely computed only at corners while at other points only partial information is available. Therefore the motion is often represented as a 2-D Gaussian random variable. The motivation for using this representation is that the Kalman filter and other methods can easily be used. However, as we show in this paper, the Gaussian approximation is valid only in special cases and often causes severe loss of data. As an alternative, we introduce a method to extract and represent displacement as a probability distribution matrix, and introduce a filter and other tools that works directly on these matrices. Using these tools, we implement a 2-D real-time tracking system. A more detailed version of this paper can be found in [5].

1 Introduction

Computing local motion is an essential stage in many computer vision tasks, such as object tracking, image registration and structure from motion. In many cases, the algorithm tracks a set of points between two or more frames. However, an exact tracking of the point's location is possible only when the point lies on a corner. In other cases, only partial information can be extracted. The prevalent solution is to represent the displacement of the feature point as a 2-D random Gaussian variable, where the covariance matrix contains the directional edge information[8][1][3]. The advantage of this representation is its simplicity and the possibility to use efficient tools such as the Kalman filter.

In [8] a Gaussian representation and a Kalman filter is used to compute an optical flow map in a video sequence. In [6] [2], Kalman filter is used for obtaining structure from motion. In [1] a 3-D tracking system is implemented using spatial and temporal filters.

In many cases, this Gaussian assumption is not valid, and leads to errors and information loss. Consider Fig. 1, two points are given and the motion between two frames is to be found. We take a small window around each point and compute the correlation between the two frames. A and B in the first row of Fig. 1 are the correlation results and in the second row, their Gaussian approximations are shown. It can be seen that the Gaussian approximation does not represent the distribution, especially in B. A consequence of trying to impose this approximation is that if the two points belongs to one scene moving uniformly and we want to find the probability distribution matrix of the whole motion using these two measurement, we should superimpose the probability matrices and normalize them. The product matrix is shown in C. It can be shown that although each point does not contains enough information about the scene's motion, we get a better idea about it from the product matrix. Trying the same with the Gaussian approximations gives the matrix C in the second row. The result is a Gaussian with a mean value of $(2, 1)$ which is far from the true motion of $(0, 0)$. This result simulates what happens when we use the Kalman filter. Thus, we can conclude that using Gaussian approximations and the Kalman filter is in many cases erroneous. Generally speaking, the Gaussian approximation is reasonable only when

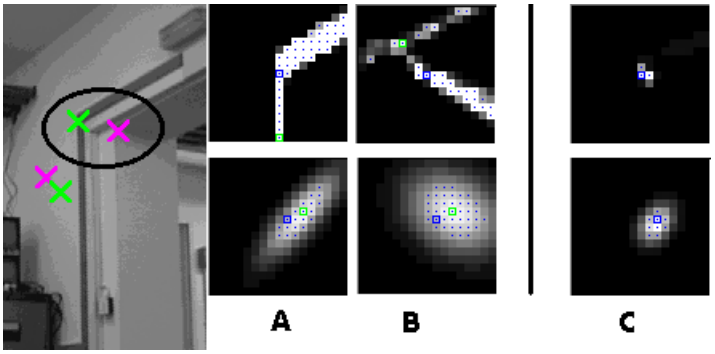

Figure 1: The probability matrices of points that cannot be approximated as Gaussians. First row: The matrices of the points within the circle (a, b). The product matrix (c). Second row: Gaussian approximation

the probability matrix is symmetric.

Another problem with Gaussian approximations concerns point tracking. When a point is not a corner, there is not enough information in the point's matrix to compute the point's displacement unambiguously. When approximating the matrix with a Gaussian, the mean is taken as the point's motion. But looking at Fig. 1,B one can see that the mean value is not a reasonable choice, because it does not coincide with a likely region of the matrix. Using this value as the point's displacement will cause the point to miss the tracked feature.

These examples demonstrate why the Gaussian approximation is not always valid. An alternative method suggested in this paper, is to represent the point's local motion as a probability distribution matrix. This will be demonstrated in a 2-D tracking system.

Tracking a set of points from an object can be used in order to track the whole object. Most methods aggregate the points motion by first making a decision about each single point, this is problematic as the single point's motion cannot usually be computed reliably without using information from all the other tracked points. Using the matrix representation, we have implemented a motion detection and tracking system which overcome these problems. The system works in real-time with live video input, even when the camera rotates. In this paper we will not describe the whole system due to space limitations. The system is used to demonstrate the matrix representation with the additional tools. This will be done by describing the main module in the tracking system.

Section 2 introduces the probability matrix formulation of a displacement and shows how a displacement's confidence can be computed from this matrix. Section 3 shows how temporal filtering can be computed directly from these matrices. Section 4 describes the computation of the object's mean motion directly from the matrices and how to compute each point's motion using this mean motion. Section 5 shows experimental results. A more detailed version of this paper can be found in [5].

2 Displacement as a probability distribution matrix

As we have shown, the displacement of a tracked point cannot be represented as a particular single value or as a Gaussian variable. The method we introduce is based on cross-SSD (sum of squared differences). The displacement is represented as a discrete probability distribution where no assumption is made about the shape of the distribution.

Formulation of the problem: Two images $\psi_1(i,j)$ and $\psi_2(i,j)$ are taken from a video sequence of a dynamic scene. Given a point $\mathbf{p}$ in ψ_1, let $\mathcal{W}$ be a window surrounding this point, and assume that all the pixels inside $\mathcal{W}$ have the same displacement as $\mathbf{p}$. We want to compute the probability distribution of the displacement of $\mathbf{p}$ between the two frames.

Let the displacement be $\mathbf{d} = (u,v)$ and let $\mathcal{W}_1$ be the window around $\mathbf{p}$ in ψ_1 and $\mathcal{W}_2$ be the window around $\mathbf{p}+\mathbf{d}$ in ψ_2. Let $P(\mathcal{W}_2 \mid \mathcal{W}_1, \mathbf{d})$ be a known function of the probability distribution of $\mathcal{W}_2$ given $\mathbf{d}$ and $\mathcal{W}_1$.

In many cases it can be assumed that the possible values for $\mathbf{d} = (u,v)$ are within the range: $U_{min}..U_{max}$, $V_{min}..V_{max}$. Let Y be defined as

$$\mathbf{Y}_{u,v} = P(\mathcal{W}_2 \mid \mathcal{W}_1, \mathbf{d} = (u,v))$$

Using Bayes' law:

$$P(\mathbf{d} \mid \mathcal{W}_1, \mathcal{W}_2) = \frac{P(\mathcal{W}_2 \mid \mathcal{W}_1, \mathbf{d})P(\mathbf{d})}{P(\mathcal{W}_2)} \qquad (1)$$

where:

$$P(\mathcal{W}_2) = \sum_{\mathbf{d}} P(\mathcal{W}_2 \mid \mathcal{W}_1, \mathbf{d})P(\mathbf{d})$$

After substituting $\mathbf{Y}_{u,v} = P(\mathcal{W}_2 \mid \mathcal{W}_1, \mathbf{d} = (u,v))$ we get:

$$P(\mathbf{d} = (u,v) \mid \mathcal{W}_1, \mathcal{W}_2) = \frac{\mathbf{Y}_{u,v}\ P(\mathbf{d} = (u,v))}{\sum_{(u,v)} \mathbf{Y}_{u,v}\ P(\mathbf{d} = (u,v))} \qquad (2)$$

$P(\mathbf{d} = (u,v))$ is the apriori probability that the displacement is $\mathbf{d}$. If no prior information is available, we take $P(\mathbf{d})$ to be constant.

It is still necessary to compute $P(\mathcal{W}_2 \mid \mathcal{W}_1, \mathbf{d})$. Let us assume that the displacement of $\mathbf{p}$ is known to be $\mathbf{d} = (u,v)$. Given the window $\mathcal{W}_1$ in the first image, the match between it and $\mathcal{W}_2$ is not perfect because of noise. The noise is generated by several sources: the camera noise, rotations, quantization errors etc. The probability distribution of the overall expected noise is very hard to compute, we present a simple method to compute the probability matrix, but other methods can work as well.

Given the displacement $\mathbf{d} = (u,v)$, the sum of squared differences of $\mathcal{W}$ between the two windows is:

$$SSD(\mathbf{d}) = \sum_{i,j \in \mathcal{W}_1} \left(\psi_2(i+\mathbf{d}_x, j+\mathbf{d}_y) - \psi_1(i,j)\right)^2$$

We model the distribution of $\mathcal{W}_2$ as a function of the SSD:

$$P(\mathcal{W}_2|\mathbf{d}) = f(SSD(\mathbf{d}), \sigma^2)$$

Assuming that the only factor that can be measured or estimated is the mean SSD value σ^2, the Maximum Entropy Criteria gives:

$$P(\mathcal{W}_2|\mathbf{d}) = c\ \exp(-SSD(\mathbf{d})/\sigma^2)$$

where c is a normalization factor.

To conclude, we have introduced a method of representing the displacement as a probability distribution matrix using SSD measurements. An example

for the probability matrices is depicted in Fig. 1. The lower row shows the Gaussian approximation of the distribution, and as can be seen, this approximation eliminates almost all the available information of the matrix.

If the motion also contains rotations, this will be interpreted as more system noise, and the probability distribution will be less sharp. However, if the rotation is not too large, the matrix still contains enough information to be useful. The effect of rotation is smaller as the window size is smaller.

3 Using temporal filtering in tracking a point

Many systems use temporal filtering to improve point tracking. When the local motion is represented as a Gaussian, the Kalman filter can be utilized. However, the Kalman filter cannot handle the probability matrix representation described above. In this section we represent an alternative filter implementation that use the probability matrix P as an input. The technique implemented here can be formalized as a filter which is a generalization of the Kalman filter for any distribution [4].

Let the **process X** be a moving point where the state $\mathbf{x}_t$ is the point's 2-D velocity vector at time t. The probability distribution of $\mathbf{x}_t$ is the matrix $\mathbf{P}_{\mathbf{x}_t}$ where each entry (u, v) is the probability that the point's velocity is $\mathbf{d} = (u, v)$. This is a square matrix with size $2\mathbf{d}_{max} + 1$. Assume for the moment that the point's motion has a constant velocity. At each time interval t, a new frame ψ_2 from the video sequence is available. We refer to it as the **measurement**. From the new frame, the matrix $\mathbf{Y}(u, v) = P(\mathcal{W}_2 \mid \mathbf{d} = (u, v))$ is computed. (see Section 2).

Using Eq. 2, the posteriori probability distribution of the process given the measurement is:

$$P^+_{\mathbf{x}_t}(u, v) = \frac{P^-_{\mathbf{x}_t}(u, v)\ P(\mathcal{W}_2 \mid \mathbf{d} = (u, v))}{P(\mathcal{W}_2)} \qquad (3)$$

where $+$ stand for posteriori distribution, and $-$ for the apriori.

Equation 3 is equivalent to computing the matrix $P^-_{\mathbf{x}_t}(u, v)\ P(\mathcal{W}_2 \mid \mathbf{d} = (u, v))$ and normalizing the matrix to give $\sum\sum P^+_{\mathbf{x}_t}(u, v) = 1$.

This simple procedure is the temporal filtering step. The problem is that it is very restrictive to assume that every point moves with a constant velocity. A better assumption is that: a) the point moves with roughly a constant velocity, and b) that this is true only for short time intervals.

The process noise. The assumption of a roughly constant velocity can be interpreted as noise in the process, that is: $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{n}$ where n is noise.

$$\mathbf{P}^-_{\mathbf{x}_t} = \mathbf{P}_{\mathbf{x}_{t-1}} \otimes P_{\mathbf{n}}$$

where $P_{\mathbf{n}}$ is the noise distribution matrix.

The filter adaptivity. Equation 3 gives equal weight for all the measurements. As it is assumed, the constant velocity assumption is valid only for short time intervals, so the filter must be adaptive. The adaptivity can be achieved by giving newer measurement more weight than older ones. An accepted method is to use exponential weighting where a measurement with an age Δt ($\Delta t = 1, 2, 3, ...$) is given a weight of $w(\Delta t) = \lambda^{\Delta t}$ [7]
Given the process current probability matrix $\mathbf{P}_{\mathbf{x}_t}$, This weighting can be implemented as follows. For each element (i, j) in $\mathbf{P}_{\mathbf{x}_t}$:

$$P'_{\mathbf{x}_t}(i, j) = P_{\mathbf{x}_t}(i, j)^{1/\lambda}$$

and normalize the matrix. λ determines the filter memory length. For example, a value of $\lambda = 1.1$ causes the seventh previous measurement to have half the weight as the new measurement. This way, the filter 'forgets' previous measurements and becomes adaptive.

An example of the filter implementation can be seen in Fig. 3. In the left column are the probability matrices before filtering, and in the middle column - after filtering. The temporal filtering sharpens the probability distribution and enhances the information about the point's motion.

4 Combining information from several points to implement 2D object tracking

Object tracking is implemented by randomly choosing N points on the object and tracking them. We do not assume that the points are on edges or corners. Besides, the object can be non-rigid and can have any motion in 3-D space. The tracking scheme implemented here is 2-D tracking, where the object's motion is defined as the **mean** translation of the N tracked points on it.

A simple but problematic way to implement such a tracking, is first to decide the motion of each of the tracked points, and then to calculate the mean motion. The problem is that the exact motion of each tracked point cannot usually be decided directly from its matrix. This is possible only for corner points, but we do not want to limit ourselves to such points.

Our method is as follows: in the first stage, the object's mean motion is calculated directly from the tracked points' filtered probability matrices. In second stage, the translation of each point is calculated using its probability matrix, and using the calculated mean object motion. (Section 4.1). In the third stage, a correction is made for each point's location in order to prevent drift. (Sections 4.2).

Let $\mathbf{P}_i$ be the distribution matrix of the i-th tracked point. Define the matrix $\mathbf{P}_{sum}$ as the sum of the distribution matrices of the N points, where the (x, y)

entry of this matrix has the value:

$$\mathbf{P}_{sum}(x,y) = \sum_{i=1}^{N} \mathbf{P}_i(x,y)$$

For each value (u,v), the expected squared error is:

$$MSE(u,v) = \frac{\sum_x \sum_y P_{sum}(x,y)[(x-u)^2 + (y-v)^2]}{\sum_x \sum_y P_{sum}(x,y)}$$

$(\bar{u},\bar{v})$ minimizes MSE, where:

$$\bar{u} = \frac{\sum_{x,y} P_{sum}(x,y)\ x}{\sum_{x,y} P_{sum}(x,y)},\ \bar{v} = \frac{\sum_{x,y} P_{sum}(x,y)\ y}{\sum_{x,y} P_{sum}(x,y)}$$

Choose $(\bar{u},\bar{v})$ as the object's motion.

Using this method it is possible to track an object by tracking some points on it even though the points are not on corners or edges. Next we will see how we can use the total motion information to compute the motion of each single point.

4.1 Updating the tracked point's location

The object's mean motion is computed directly from the probability distribution matrices of the tracked points. However, in order to track the points to the next frame, the point's displacement needs to be computed. This can be done locally only for corner points. In our tracking system we choose the motion of every point so that it is as close as possible to the object's mean motion.

The method is first to find the set of entries for the matrix $\mathbf{P}_{\mathbf{x}_t}$ that satisfy:

$$S = \{\mathbf{s} \in S \ \mid \ P_{\mathbf{x}_t}(\mathbf{s}) / P_{\mathbf{x}_t\ max} \geq \epsilon\} \qquad (4)$$

and from this set choose the entry $\mathbf{s} = (u,v)$ with minimal size of $(u-\bar{u})^2 + (v-\bar{v})^2$ as the displacement $\mathbf{d}$.

4.2 The drift problem

With the method described here the local motion is computed up to one pixel accuracy. This can cause the tracked points to drift from its initial position after a few frames. In order to prevent drift we need a computation method whose *mean* computation error over a long sequence of frames is always zero, but can still have instantaneous errors. In [5] we show how to achieve such an accuracy using entropy measurements on the point's distribution matrices.

5 Experimental results

We have implemented the tracking system described in the above sections. Before the tracking, twenty points were randomly chosen on the tracked object. Notice that most of the points are not on edges, and therefore are more difficult to track. The tracking was carried out for forty frames. The video was taken with a moving camera, so that both the object and the background are moving. In Fig. 2, eight out of the forty frames are depicted. It can be seen that most of the points where successfully tracked through the sequence.

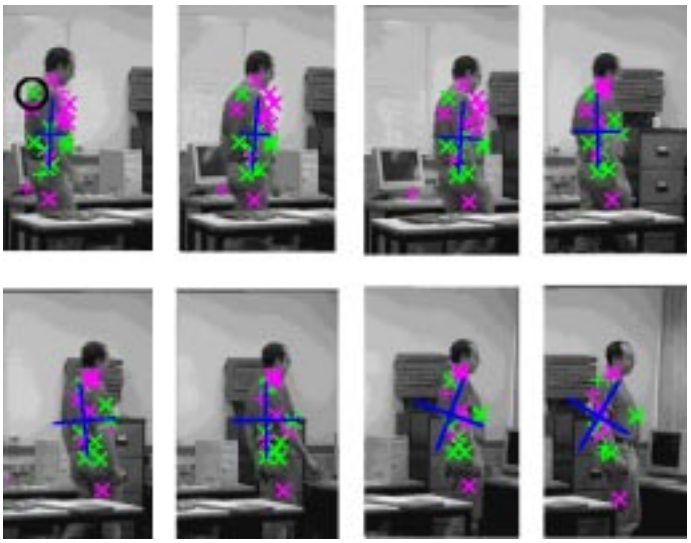

Figure 2: Tracking a moving person with a moving camera. Eight frames are shown out of forty. Points marked as X detect motion. The big cross is the points' center of mass.

In the figure, points marked as X are those where motion was detected, and the points marked as $+$, are where no motion was detected. The motion detector is implemented very simply by checking the probability of the entry $(0,0)$ in the process's probability matrix $\mathbf{P}_{\mathbf{x}_t}$.

The process's noise distribution matrix $P_{\mathbf{n}}$ was chosen as $\mathcal{N}(0, 0.75)$. The adaptivity factor is $k = 0.7$, The matrices size are $(-8..8, -8..8)$ so that translation up to 8 pixels between frames can be detected. the window size was 5×5 pixels. In Fig. 3, the probability matrices are depicted. The left column is a sequence of measurement matrices $\mathbf{Y}$ belonging to the tracked point marked with a circle in Fig. 2. It can be seen that the distributions are not always similar to a Gaussian. The middle column is the probability matrix of the process $\mathbf{P}_{\mathbf{x}_t}$, i.e. the motion distribution after filtering. The filter effect can be seen as the probabilities are sharper than in the instantaneous measurement matrix.

In the right column, the sequence of the sum matrices $\mathbf{P}_{sum}(x,y)$, which represents the motion of all the points is shown. The matrices' entries with a point inside represents entries with probability close to maximum, i.e. the entries belongs the set S. The motion of each tracked point is chosen from this set as the one closest to the object's mean motion, as discussed in 4.1.

This example demonstrates how by using the tools developed in this paper, a real-time tracking can be implemented relatively easily. Tracking is achieved

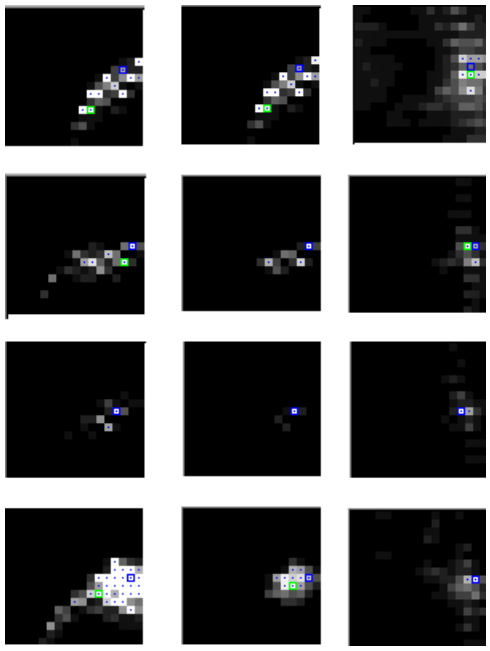

Figure 3: The probability matrices of the first four frames. The current measurement between the last two frames (left), after temporal filtering (middle) and the sum matrix from all the tracked points (right).

with only a small number of tracked points, without feature detection. It works even when the object cannot be considered rigid, (for example - the hand motion which is different than the body motion), and when the object has rotation. The tracking works well with live video on a PC in real-time.

For comparison, we have tried to repeat the tracking when we approximate the matrices with Gaussians and using the Kalman filter. The results are described in [5].

6 Summary

It has been shown that local motion cannot be represented as a Gaussian and that a better approach is to represent it with a distribution matrix. A filter and other tools for manipulating these matrices are derived. A 2-D tracking system was implemented using these tools. The distribution matrix representation allowed us to implement the tracking of the whole object without forcing us to make a prior decisions about single points which made the system much more robust.

We believe that the matrix representation can be used to solve other computer-vision tasks such as optical flow, image registration, structure from motion and 3-D tracking.

References

[1] D. B. Gennery. Visual tracking of known three-dimensional objects. *International Journal of Computer Vision*, pages 243–270, 1992.

[2] R. Szeliski L. Matthies, T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *IJCV*, (3):209–236, 1989.

[3] I.D. Reid and D.W. Murray. Tracking foveated corner clusters using affine structure. *International Conf. on Computer Vision*, pages 76–83, 1993.

[4] Y. Rosenberg and M. Werman. A general disribution filter for measurements with any probability distribution. *CVPR 1997*, page http://www.cs.huji.ac.il/papers/IP/CVPR97/index.html.

[5] Y. Rosenberg and M. Werman. Representing local motion as a probability distribution matrix for object tracking and other applications. *http://www.cs.huji.ac.il/papers/IP/CVPR97/index.html.*

[6] A. Shmuel and M. Werman. Active vision: 3d from an image sequence. *Tenth International Conference on Pattern Recognition*, A:48–54, 1990.

[7] A. Singh. An estimation-theoretic framework for image-flow computation. *International Conf. on Computer Vision*, pages 168–177, 1990.

[8] A Singh. Incremental estimation of image-flow using a kalman filter. *Workshop on Visual Motion*, pages 36–43, 1991.