MVA2013 IAPR International Conference on Machine Vision Applications, May 20-23, 2013, Kyoto, JAPAN

3D Keypoint Tracking Based on Hybrid Flow Computation for Human Motion Analysis

Sujung Bae, Sungeun Hong, Hyun S. Yang
Korea Advanced Institute of Science and Technology (KAIST)
291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea
{sjbae, fastacon, yang}@paradise.kaist.ac.kr

Abstract

In this paper, we propose a robust method to track 3D keypoints for representing 3D human motions under noisy depth environment. Once keypoints are extracted for tracking, the subsequent locations of the keypoints are found by estimating their 3D motion flows. However, estimating the flows using depth information would be a problem, especially, when some depth values are unavailable or the data points are severely affected by noise. To handle such circumstances, we suggest hybrid flow computation scheme that selectively utilizes range flow and optical flow estimation methods. Experimental results show that the proposed method outperforms the conventional flow estimation methods under problematic environment in terms of tracking accuracy and performance, with linear increase in computational time.

1 Introduction

Human motion analysis has been a key technology in many future-oriented applications in the field of human-computer interaction (HCI), visual surveillance, and home healthcare. For such reasons, it has been vigorously researched in the last two decades.

Human motion can be basically represented in 2D and 3D. Although approaches of representing human motion in 2D have low complexity and short computation time, they have depth ambiguity and thus can not completely represent 3D motions of humans. Recently, there has been new efforts to devise 3D-based features that reflect actual human motions[1, 2]. There are mainly two ways of reflecting 3D human information in studies. The first method constructs the human body in forms of 3D volumes and skeletons using multiple cameras. The other method constructs the depth information from an entire scene with human objects. The depths are estimated by either stereo vision or range-camera technology. However, these studies have limitations in that their tasks are time-consuming with high complexity and cost, compared to 2D-based tasks. Fortunately, with the development of Kinect, depth information can be obtained quickly with low cost and complexity. The advent of Kinect opened novel possibilities for devising competitive 3D-based features that reflect actual human motions.

To represent motion in 2D by tracking keypoints from an image, optical flow estimation method is used in many cases[2]. For 3D cases, in correspondance with 2D, range flow estimation method can be used for representing 3D motion[4]. As stated in [5], where range-flow field is used to represent the motion field of a scene, manipulation of depth values obtained via Kinect raises some issues: i) the depths are not always stable; ii) there are some areas in which the depth values are not available. These problems worsen especially in the contour areas of humans and objects.

In this paper, we propose a novel hybrid flow computation scheme for tracking 3D keypoints to represent human motions. Once keypoints are extracted in human-body areas, the points are tracked in a pyramidal tracking framework. The subsequent locations of keypoints are found by using 3D motion flow estimation. When depth values for the keypoint are sufficiently stable, range flow is estimated by applying the algorithm suggested by Barron and Spies[4]. In case the depth values are unavailable or severely affected by noise with increased fluctuation, optical flow algorithm suggested by Horn and Schunck[3] is used instead for keypoint tracking. This makes our proposed method robust under problematic depth environment. Thus, our main contribution is in the development of a robust 3D keypoint tracking method to handle data with problematic depths, which are typically generated by low-cost devices.

2 Preprocessing

Two preprocessing steps are necessary for tracking motion with color and depth images obtained through Kinect. First, the depth image needs to be in alignment with the color image, because color and infrared (IR) cameras of Kinect are placed in different location and their fields-of-view (FOVs) are different. For the alignment, we use a function provided by OpenNI library[8]. Then, we conduct normalized convolution (NC) on the depth image to reduce those areas in which the depth values of pixels are not available. This unavailability is the result of limitation of Kinect in computing depth. The NC estimates unknown values by interpolation of known values[7]. Gaussian filter is used as the smoothing filter for NC.

3 3D Keypoint Tracking

3.1 Keypoint selection

The keypoints to be tracked need to be selected before the tracking phase. As stated in [6], a good point to be tracked satisfies equation (1) where λ_1 and λ_2 are two eigenvalues of matrix

$$\mathbf{G} = \begin{bmatrix} \sum I_{x^2} & \sum I_x I_y \\ \sum I_x I_y & \sum I_{y^2} \end{bmatrix}$$

$$min(\lambda_1, \lambda_2) > thresh \qquad (1)$$

An element of $\mathbf{G}$ is composed of the sum of gradients about x and y directions in a grayscale image, denoted by I_x and I_y, on a small patch centered at a point.

Among the selected points, those outside the human area are not of interest, so they are discarded. To do that, we receive help from a function performing human area segmentation utilizing depth information in NITE libraries[9].

3.2 Keypoint tracking

A selected point in 2D and its corresponding depth value constitute a projective point $p = [x, y, Z]^T$. We do not convert it to a point $P = [X, Y, Z]^T$ in *Camera Space*, which is 3D space centered at the camera. This is because it is easy to compute gradients of intensities and depths for each point in the projective space. However, for notation consistency when explaining the conventional flow estimation algorithms, a projective point $p = [x, y, Z]^T$ is denoted by $P = [X, Y, Z]^T$.

The keypoint tracking problem and its solution are described as follows. Let $p = [X, Y, Z]^T$ be a keypoint to be tracked at time t. The problem is to find p' which is the location of p at time $t + 1$. We solve the problem by deriving p's displacement f, 3D motion, in time between t and $t + 1$, which is formulated by $p' = p + f$.

Algorithm 1 Pyramidal 3D keypoint tracking

```
Require: $L_{MAX}$, $L_{MIN}$, $\mathbf{I}$, $\mathbf{J}$, $\mathbf{ZI}$, $\mathbf{ZJ}$, $\mathbf{P}$
Ensure: $\mathbf{P}'$, $\mathbf{F}$
1: Normalize depth images, $\mathbf{ZI}, \mathbf{ZJ}$
2: Build pyramidal images for $\mathbf{I}, \mathbf{J}, \mathbf{ZI}, \mathbf{ZJ}$
3: for pyramid level $l = L_{MAX}$ to $L_{MIN}$ do
4:     Smooth $\mathbf{I}^l, \mathbf{J}^l, \mathbf{ZI}^l, \mathbf{ZJ}^l$
5:     Adjust $\mathbf{P}^l, \mathbf{ZI}^l, \mathbf{ZJ}^l$ for level $l$
6:     Compute flows $\mathbf{F}^l$ of $\mathbf{P}^l$ with Algorithm 2
7: end for
8: $\mathbf{F} \leftarrow \mathbf{F}^{L_{MIN}}$
9: $\mathbf{P}' \leftarrow \mathbf{P} + \mathbf{F}$
```

We adopt a pyramid representation to handle large motions, which is a classical approach used in 2D keypoint tracking. Algorithm 1 describes the overall pyramidal 3D keypoint tracking framework. In the algorithm, let $\mathbf{I}$ and $\mathbf{J}$ be the grayscale images, and $\mathbf{ZI}$ and $\mathbf{ZJ}$ be the depth images at time t and $t + 1$, respectively. $\mathbf{P}$ is a set of keypoints at time t. L_{MAX} and L_{MIN} denote the maximum and minimum pyramidal levels. The algorithm starts by normalizing depth values and adjusting the scale to that of the intensity. All variables with superscript l denote their values at level l. For example, $\mathbf{I}^l$ is an image at level l. We build pyramidal images so that the sizes of images at level $l - 1$ are twice as large as those of the images at level l. At each pyramid level, we need to relocate points and readjust depth values according to that level after smoothing images, which are

$$\begin{aligned} &\mathbf{P}^l = \mathbf{P}/2^l \\ &\mathbf{ZI}^l = \mathbf{ZI}/2^l, \mathbf{ZJ}^l = \mathbf{ZJ}/2^l \end{aligned} \tag{2}$$

Then, $\mathbf{P}^l$'s flow $\mathbf{F}^l$ is computed with Algorithm 2 which will be explained next. When the last level of the pyramid is reached, the flows derived at that level become $\mathbf{P}$'s displacement.

The range flow estimation is suggested by Barron and Spies to represent 3D motion for a point on surface[4]. The flow estimation problem is described as in (3) where both motion constraints and the partial derivatives of the flow are to be minimized. In the expression, (Z_X, Z_Y, and Z_t) and (I_X, I_Y, and I_t) are the depth and intensity gradients about positions X and Y, and time t. Further, ($[U_X, U_Y, U_Z]$, $[V_X, V_Y, V_Z]$, and $[W_X, W_Y, W_Z]$) denote partial derivatives for each of the flow components U, V, and W with respect to positions X, Y, and Z.

$$\begin{aligned} &\int\int\int\int (Z_X U + Z_Y U + W + Z_t)^2 \\ &\quad +\beta^2 (I_X U + I_Y V + I_t)^2 \\ &\quad +\alpha^2 (U_X^2 + U_Y^2 + U_Z^2 + V_X^2 + V_Y^2 + V_Z^2 \\ &\quad +W_X^2 + W_Y^2 + W_Z^2) dX dY dZ dt \end{aligned} \tag{3}$$

The solution for equation (3) is

$$\begin{aligned} &\begin{bmatrix} U^{k+1} \\ V^{k+1} \\ W^{k+1} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} \alpha^2 \overline{U}^k - Z_X Z_t - \beta^2 I_X I_t \\ \alpha^2 \overline{V}^k - Z_Y Z_t - \beta^2 I_Y I_t \\ \alpha^2 \overline{W}^k - Z_t - \beta^2 I_t \end{bmatrix}, \\ &\mathbf{A} = \begin{bmatrix} Z_X^2 + \beta^2 I_X^2 + \alpha^2 & Z_X Z_Y + \beta^2 I_X I_Y & Z_X \\ Z_X Z_Y + \beta^2 I_X I_Y & Z_Y^2 + \beta^2 I_Y^2 + \alpha^2 & Z_Y \\ Z_X & Z_Y & 1 + \alpha^2 \end{bmatrix} \end{aligned} \tag{4}$$

where $[U, V, W]^T$ is the range flow, and $[\overline{U}, \overline{V}, \overline{W}]^T$ is the local average reflecting neighboring flows estimated in the previous iteration. The solution is very similar to Horn and Schunck's iterative optical flow solution[3], as shown below.

$$\begin{aligned} &\begin{bmatrix} u^{k+1} \\ v^{k+1} \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} \alpha^2 \overline{u}^k - I_x I_t \\ \alpha^2 \overline{v}^k - I_y I_t \end{bmatrix}, \\ &\mathbf{A} = \begin{bmatrix} I_x^2 + \alpha^2 & I_x I_y \\ I_x I_y & I_y^2 + \alpha^2 \end{bmatrix} \end{aligned} \tag{5}$$

where $[u, v]^T$ is the optical flow and $[\overline{u}, \overline{v}]^T$ is the local average. The solution for the optical flow will be calculated when we can not estimate range flow.

Algorithm 2 Iterative estimation of flow

```
Require: $p^l$, $gf$, $l$, $L_{MAX}$, $L_{MIN}$, $\mathbf{I^l}$, $\mathbf{J^l}$, $\mathbf{ZI^l}$, $\mathbf{ZJ^l}$
Ensure: $f^l$, $gf$
1: if $l = L_{MAX}$ then
2:     Initialize flow guess $gf^l$ with zeros
3: end if
4: Make patches of $p^l$ with $gf^l$
5: if All the depth values in patches are available then
6:     Iteratively compute range flow $f^l$
7: end if
8: if There are unavailable depths in patches, or Line 6
   is not converged then
9:     Iteratively compute optical flow $f^l$
10:    component $W$ of $f^l \leftarrow 0$
11: end if
12: if $l \neq L_{MIN}$ then
13:    $gf^{l-1} \leftarrow 2(gf^l + f)$
14: else
15:    $f^l \leftarrow gf^l + f^l$
16: end if
```

More details about Algorithm 2 is given for a keypoint to be tracked. We limit neighboring points and their flows to those defined on a small patch around the keypoint. Therefore, we ensure that the patches of the keypoint correspond to $\mathbf{I}, \mathbf{J}, \mathbf{ZI}$, and $\mathbf{ZJ}$. A patch ranges from $Y - WIN_Y$ to $Y + WIN_Y$ for the y-axis and from $X - WIN_X$ to $X + WIN_X$ for the x-axis. The WIN_Y and WIN_X denote the patch's height and width. The patches on $\mathbf{J}$ and $\mathbf{ZJ}$ are translated by *flow guess* at level l which is the flow computed at a previous level for the keypoint. Then we compute flows for all the points on a patch by appying the aforementioned iterative solution for range flow. Later, only the center point on a patch has a chance to supply a flow for the keypoint. Although we attempt to remove those pixels whose depth values are unavailable, as described in Section 2, we can encounter such pixels on depth patches. In addition, it is possible that the iteration for the solution may not converge within the limited number of times owing to severely noisy depths. For such a point, we compute the optical flow instead of the range flow and apply equation (5) instead of equation (4). When calculating the solutions, for computing gradients we follow a method suggested by Horn and Schunck[3]. However, the manner of computing local averages is slightly different from that suggested by Horn and Schunck. In our algorithm, local averages are computed through two-step convolution of the neighboring flows with weights $\mathbf{W_1}$ and $\mathbf{W_2}$. Equation (6) describes this convolution for a flow component U.

$$\overline{\mathbf{U}}^k = (\mathbf{F}_U^{k-1} \otimes \mathbf{W_1}) \otimes \mathbf{W_2} \tag{6}$$

where $\mathbf{F}_U^{k-1}$ is a matrix having values of the flow component U obtained from the previous iteration on the patch. $\mathbf{W_1}$ is a Gaussian kernel having the same size as the patch, and $\mathbf{W_2}$ is the matrix

$$\begin{bmatrix} 1/12 & 1/6 & 1/12 \\ 1/6 & 0 & 1/6 \\ 1/12 & 1/6 & 1/12 \end{bmatrix}$$

The averages for $\overline{\mathbf{V}}^k, \overline{\mathbf{W}}^k, \overline{\mathbf{u}}^k$, and $\overline{\mathbf{v}}^k$ are computed in the same manner.

Once the iteration is terminated, we obtain the estimated flows on the patch. Then, a point p's flow f at that pyramidal level becomes the flow at the center of the patch. This flow is readjusted and becomes *flow guess* for the next pyramidal level. If the current level is the last level, the flow becomes the final flow along with *flow guess* at that level. Algorithm 2 briefly describes the aforementioned process.

3.3 Additional tasks

During tracking, keypoints are lost owing to several reasons: i) a point exits in the image, ii) either 2D or 3D norm of a flow is too large, iii) patches centered at a tracked p' on J and ZJ are very different from those centered at p on I and ZI. In these cases, we consider a point as lost and discard it. When there are an insufficient number of keypoints at frame t, additional keypoints are newly selected and added.

In addition, since we scale down the depth values for stable tracking, a tracked Z position tends to be less accurate than tracked X and Y positions. To increase its accuracy, we alpha-blend the value of tracked Z with the depth value corresponding to the tracked (X, Y) position in ZJ.

4 Experimental Results

Two different tracking results are compared to our tracking result. The first set of results are obtained from a tracking method using only range flow, referred to as M2. The second set of results are obtained from a tracking method using 3D optical flow, referred to as M3. In M3, keypoints are tracked using optical flow in 2D. Then, depth values at the tracked locations become the points' Z positions. Hereafter, we denote our tracking method by M1. The projective points from M1, M2, and M3 are converted to points in *Camera Space* to represent the points in 3D space. Units of three axes for *Camera space* are millimeters. Kinect produces videos at 30 frames per second (FPS). We recorded six movies containing three simple activities performed by two subjects in front of Kinect. Table 1 gives descriptions of these activities. In the table, # denotes the total number of frames of the movies of two subjects for a activity. While tracking the points, if the number of tracked points are less than 250, we add newly selected points.

Table 1. Descriptions of activities

Activity	#	Description
1	255	A subject moves forward and backward
2	298	A subject make a gesture of pushing and pulling with his/her arm
3	505	A subject moves his/her body from side to side

We compare M1, M2, and M3 in terms of tracking accuracy and performance. There is no dataset with ground truth recorded by Kinect to measure the accuracy as the unit of a point. Therefore, the comparison is examined in statistical point of view. In terms of the tracking performance, we focus on how many points are tracked and for how long the points are tracked.

The comparison results are presented in Table 2. In the table, the first three rows show averages and standard deviations about the X,Y, and Z locations of tracked points. The next three rows are concerned with the three flow components. In each of the six rows, standard deviations are described in parentheses. The row starting with NT shows the total number of tracked keypoints.

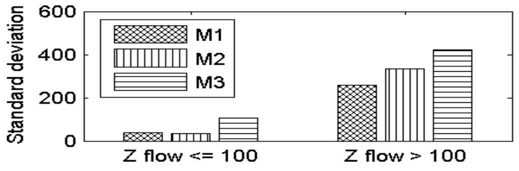

Figure 1. Standard deviations of two flow groups.

The comparison results show two noteworthy points. First, M1 tends to produce smaller standard deviations especially in the third component of flow, labeled as Z flow, than the other methods. The large variances of M2 and M3 are caused by erroneous Z flows, as shown in Figure 1. The figure illustrates the standard deviations of two groups where Z flows are grouped as normal (Z flow $<= 100$ mm) and erroneous (Z flow $>$

Table 2. Comparison results

	Activity 1			Activity 2			Activity 3		
	M1	M2	M3	M1	M2	M3	M1	M2	M3
X	113.5(160.7)	111.4(201.9)	109.2(159.4)	43.7(171.0)	44.3(204.0)	33.9(163.7)	132.8(355.8)	156.5(418.4)	124.1(359.9)
Y	-156.8(311.3)	-123.2(347.7)	-150.0(310.4)	-173.5(427.8)	-177.5(433.3)	-186.7(420.6)	-166.6(391.7)	-142.3(455.0)	-157.6(381.3)
Z	1607.7(603.7)	1682.1(848.3)	1593.1(546.9)	2096.1(657.5)	2143.1(857.1)	2075.8(590.6)	2268.2(466.0)	2435.1(870.9)	2265.4(465.5)
X Flow	-0.0(7.7)	-0.0(6.9)	-0.0(17.2)	0.4(10.5)	0.8(9.6)	0.1(18.8)	0.2(15.4)	0.4(13.2)	0.2(30.5)
Y Flow	-0.1(9.5)	-0.2(10.3)	-0.1(24.6)	-0.1(10.7)	0.2(9.3)	-0.1(32.4)	-0.0(11.0)	0.0(11.4)	-0.1(28.1)
Z Flow	3.3(60.8)	6.6(76.8)	2.6(130.8)	2.6(76.3)	5.6(85.6)	1.1(169.8)	1.9(66.5)	2.7(67.6)	2.2(159.3)
NT	61381	22914	60481	63709	39323	62460	124202	31843	118323

100 mm). In the figure, M2 and M3's erroneous flow groups produce much larger standard deviation than M1's. Second, M1 tracks more keypoints for a longer time than M2 and M3, which is shown in the last row in Table 2 and Figure 2. During the entire time frame, we accumulate the number of tracked points which are the points that survive without failure within each frame. The accumulated number for each method is described in the row labeled with NT in Table 2. According to this, the points in M1 are tracked 2.65 times more than the points in M2, while the difference in the numbers of tracked points between M1 and M3 is negligible. Through the comparisons, it turns out that M1 has fewer occurrences of lost points than the other methods while tracking keypoints. This also affects the lifetime of tracked points, indicated by the number of frames from the beginning of tracking to the failure of tracking. Each tracked point's lifetime is examined and the result is illustrated in Figure 2. The two numbers in

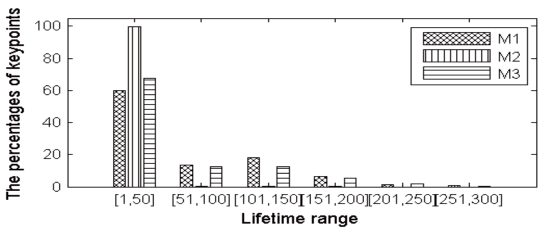

Figure 2. Percentage of points belonging to a specific range of lifetime.

square brackets on the x-axis indicate the beginning and end of the lifetime range. As shown in the figure, the points tracked by M1 have the highest possibility to survive for a long time. On the other hand, the points tracked by M2 have the highest possibility to be lost within a short time. Figure 3 illustrates the visualization of tracked 3D keypoints and their flows for a scene where the subject pushes with her arm. The keypoints and flows are drawn as black points and magenta lines, respectively. The blue lines and red rectangles, respectively, indicate the subject's skeletons and 15 joints whose locations are obtained by NITE library[9].

5 Conclusion

Thus far, we have looked at 3D keypoint tracking method by estimating 3D motion flow under problematic depth environment. Through experiments, we found that our proposed hybrid flow computation scheme makes the keypoint tracking robust, compared to the other methods just using conventional flow such as optical and range flow. The proposed method requires slightly longer computation time than the other methods. Nevertheless, we expect this method is beneficial to researchers who attempt to understand human behavior under problematic environment.

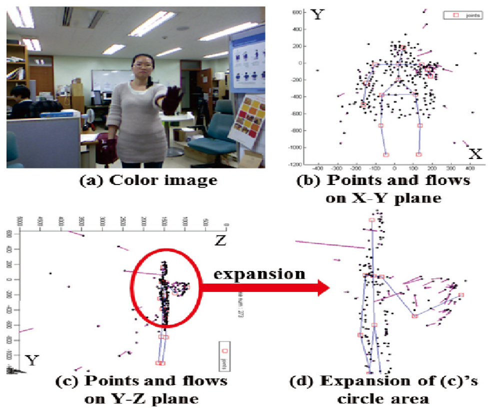
Y
X
(a) Color image
(b) Points and flows on X-Y plane
Z
Y
expansion
(c) Points and flows on Y-Z plane
(d) Expansion of (c)'s circle area

Figure 3. Visualization of 3D keypoints and their flows.

Acknowledgments: This research was supported by the IT R&D program of MKE/KEIT, (10039165, Development of learner-participatory and interactive 3D virtual learning contents technology)

References

[1] T. B. Moeslund, A. Hilton, V. Krger: "A survey of advances in vision-based human motion capture and analysis," *Computer Vision and Image Understanding*, vol.104, pp.90-126, 2006.
[2] X. Ji, H. Liu: "Advances in View-Invariant Human Motion Analysis: A Review," *IEEE Trans. Systems, Man, and Cybernetics*, vol.40, no.1, pp.13-24, 2010.
[3] B. K. P. Horn, B. G. Schunck: "Determining optical flow," *Artificial Intelligence*, vol.17, pp.185-203, 1981.
[4] J. L. Barron, H. Spies: "The Fusion of Image and Range Flow," *Multi-Image Analysis*, vol.2032, pp.171-189, 2001.
[5] J.M. Gottfried, J. Fehr, C. S. Garbe: "Computing Range Flow from Multi-modal Kinect Data," *International Symposium on Visual Computing*, 2011.
[6] J. Shi, C. Tomasi: "Good features to track," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp.593-600, 1994.
[7] Next Generation Artificial Vision Systems, *Artech House*, 2008.
[8] OpenNI Library: `http://openni.org/`
[9] NITE Library: `http://www.primesense.com/nite`