Motion Estimation from RGB-D Images Using Graph Homomorphism

David da Silva Pires[1], Roberto M. Cesar-Jr[1], and Luiz Velho[2]

[1] University of São Paulo, São Paulo, Brazil
{davidsp,cesar}@vision.ime.usp.br
http://www.vision.ime.usp.br/creativision

[2] National Institute for Pure and Applied Mathematics, Rio de Janeiro, Brazil
http://www.visgraf.impa.br

Abstract. We present an approach for motion estimation from videos captured by depth-sensing cameras. Our method uses the technique of graph matching to find groups of pixels that move to the same direction in subsequent frames. In order to choose the best matching for each patch, we minimize a cost function that accounts for distances on RGB and XYZ spaces. Our application runs at real-time rates for low resolution images and has shown to be a convenient framework to deal with input data generated by the new depth-sensing devices. The results show clearly the advantage obtained in the use of RGB-D images over RGB images.

Keywords: motion estimation, graph matching, RGB-D images.

1 Introduction

With the advent of devices like Kinect™ (from Microsoft®) and Xtion™ (from ASUS®) that capture texture and depth images from a scene, there are many new challenges and problems to be faced. One of the main applications for data captured by such equipments is generally concerned with natural interaction. These applications typically use anthropometric algorithms to estimate pose, skeleton and the number of users in front of the device. Some systems have specialized algorithms to recognize its users, even if there is identical twins among them [1]. Gesture recognition using Kinect™ has been used as a control to other devices, aiming an easier or more natural interaction and allowing the use of computers with great accessibility [2].

1.1 Objective and Motivation

This work aims to show the benefits of using the additional information given by the depth image registered with a texture image, presenting an algorithm, based on graph matching, that detects the direction of movement at real-time rates. The developed procedure shows, with labels identified by colors or, optionally, with the use of arrows, to which direction each group of pixels (rectangular areas, called patches, arranged in a regular grid on the image) is moving.

© Springer-Verlag Berlin Heidelberg 2013

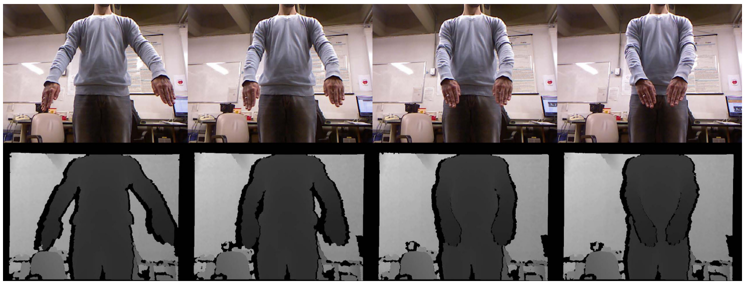

Fig. 1. Input data is a video sequence of texture and depth map images per frame, providing the RGB and (x, y, z) values of each pixel

Depth data, when added to the traditional RGB values and texture coordinates, help the delimitation of objects of interest and makes results substantially better when considered as a descriptive feature for each pixel. This characteristic is an advantage with respect to methods that depend on the presence of a well defined pattern on texture, like checkerboard sequences [3].

The technique developed on this paper may be useful on general applications, such as 3D scene reconstruction and augmented reality. Our approach is also an intermediary step to identify the rigid components of an articulated object [4].

Given a video sequence, such as the example shown at Figure 1, the application builds a graph for each frame and compare them subsequently, finding a matching based on distances at RGB and XYZ spaces.

2 Related Work

The evolving technology regarding depth-sensing devices was initially created to provide natural interaction to video-games. However, recent Computer Vision and Graphics research has shown a lot of other interesting uses.

The Kinect Identity technology [1] explores a set of three independent identification techniques: face recognition, clothing color tracking and height estimation. These techniques were selected from a major set and demonstrated to be the best ones that, at the same time, were robust, non-CPU and memory intensive and as independent as possible from each other. Such choice indicates the importance of the development of tracking algorithms that uses both kinds of data: texture and depth map.

The lack of a better treatment of the depth data in conjunction with the RGB data and, consequently, the use of them on motion estimation algorithms is felt even by developers of software specialized in gesture recognition, like FAAST, the Flexible Action and Articulated Skeleton Toolkit [5]. They have demonstrated interest in developing real-time head tracking and estimation of the twist of the user's arm. None of these are provided by the middle-ware OpenNI™ and the solutions to these problems certainly involves Computer Vision techniques to be applied to both input data.

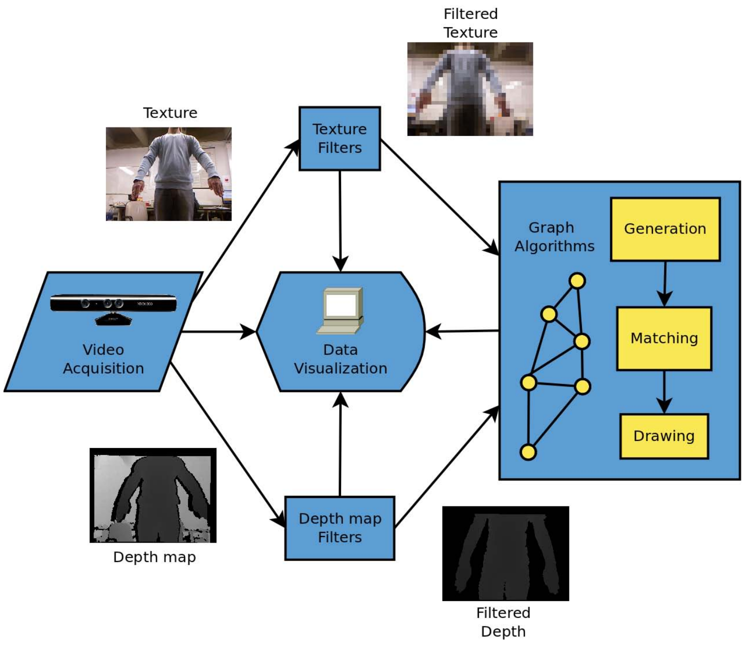

Fig. 2. Data flow representing the implemented method

In the present work, we use graph matching to find a correspondence between two point sets. This approach has been used to solve many Computer Vision problems such as interactive natural image segmentation [6], computer-assisted colorization [7] and point matching for non-rigid registration [8], among others [9].

3 Methodology

Our method is a kind of discrete optimization for determining optical flow. The data processing occurs according to a specific pipeline that is composed of the following steps: data acquisition; texture filters; depth map filters; graph algorithms; data visualization. The data flow is schematized in Figure 2.

3.1 Graph Based Approach

In order to find a matching, we have to consider relevant features that describe the points and to have a way to compare such features. Thus, each frame generates a graph whose vertices are derived from patches properties. We used six values for each pixel on the input data: RGB data, extracted from color channels, and (x, y, z) data, with x and y being texture coordinates and z being the distance to the capturing device. The frame representation is given by an attributed relational graph (ARG), allowing storage and comparison of structural, temporal and quantitative information. The recognition of the direction of the movement is done through an inexact graph matching. This approach allows differences between model and input graph [10]. In the present paper, each pair of subsequent frames generates the model and input graphs.

An ARG is a graph whose vertices represent objects while edges denote relations among them. Objects can be characterized by a finite number of attributes (numerical or symbolic), such as area, perimeter, color and shape. The relations often correspond to distances and relative orientation between objects, although more rich spatial relations may be adopted. With the contents of each frame being represented by an ARG, motion estimation resumes to a graph matching, consisting of a determination of a mapping of the vertex set of an ARG to the vertex set of another one.

Each graph is treated as a complete graph, in the sense that every vertex is connected to all the other vertices. We compute a cost function (see Section 3.3) involving the distance between the RGB and depth values of each pair of vertices: (v_m, v_i), where v_m and v_i are vertices from the model and the input graphs, respectively. The pair that minimizes this cost function is added to the matching set.

While graph vertices store point sets, including position information about these points, structural relations are stored at graph edges.

3.2 Graph Generation

The model and the input graph are built from consecutive pairs of texture and depth map input frames. Thus, at the beginning of the acquisition procedure, we can build just one graph. As the subsequent frames are captured, the input graph relative to the immediate past frame is assigned to the model graph and a new input graph is built from the new data acquired.

In order to build the graph, we consider the representation of the input images (texture and depth map) composed by patches. Given an image and the patches' parameters, we can compute how much patches compose the new representation, being sufficient to make a division between the number of rows and columns of each one. Thus, patches' dimensions are directly related with the size of the graphs that are created, highly influencing the performance of the application.

Each patch is a candidate to have a vertex representing it on the graph, being elected based on its z (depth) value. A new vertex is created and inserted on the graph only if its z value do not belong to shadow areas on depth map or if it is not too close nor too far of the capturing device. This perspective treats the depth map as a valid mask to texture pixels and allows easy background elimination based on a predefined threshold.

3.3 Graph Matching

The matching is done between two graphs: model and input. The model graph represents the last pair of frames (texture and depth images), captured before the current one, which is represented by the input graph.

A matching is an ordered pair of vertex descriptors, the first one referring to a model graph vertex and the second one relative to an input graph vertex.

For each vertex belonging to the model graph, we find the vertex on the input graph that minimizes the cost function. Eventually, the matched vertices have the same (x, y) texture coordinates, indicating that no movement has occurred at that location between the two pairs of frames.

The cost function c is given by a convex combination of two distances, d_{RGB} and d_{XYZ}:

$$c = \alpha \cdot d_{RGB} + (1 - \alpha) \cdot d_{XYZ}. \tag{1}$$

The d_{RGB} value measures the distance of the color of the patches being compared on RGB space, while the d_{XYZ} value measures the distance between the (x, y) texture coordinates and between the z depth values. As it can be seen, the parameter α controls how much each distance is considered at the final value of the cost function.

When calculating the cost function, we need to decide about which distance function to use. Two different distances have been implemented, the city block (Manhattan):

$$d_{RGB} = \left|v_R^M - v_R^I\right| + \left|v_G^M - v_G^I\right| + \left|v_B^M - v_B^I\right|, \tag{2}$$

and the Euclidean distance:

$$\begin{aligned} d_{RGB} &= \left\|v_{RGB}^M - v_{RGB}^I\right\|_2 \\ &= \left\|\left(v_R^M - v_R^I, v_G^M - v_G^I, v_B^M - v_B^I\right)\right\|_2, \end{aligned} \tag{3}$$

where the raised indexes M and I indicate if the vertex belongs to the model or input graph, respectively, while the sub-indexes R, G and B indicate the channel color being considered. The same formulas are applied to the (x, y, z) depth map values. Best results were achieved with the use of Euclidean distance.

4 Results and Discussion

The present section illustrates the results produced by the system with some real cases. All the examples were captured at 30 fps, with VGA resolution (640×480) and run in an Intel Core 2 duo computer. Figure 3 shows the captured depth of two subjects walking at opposite directions, with occlusion occurring between them and also between their respective legs. The image at Figure 3(b) shows the same depth at Figure 3(a) after background elimination filtering. Figure 3(c) shows the texture for the same scene. Note that the depth is already registered with the texture. Finally, Figure 3(d) shows the detected motion represented as color labels, with the colors red, green and blue representing movement to right, up and left directions, respectively. As we can see, the method accounts for the effects of occlusion. This experiment shows the identification of motion present on the scene. The leftmost subject is more distant to the capturing device, as indicated by the gray levels in the depth map; it is walking from left to right.

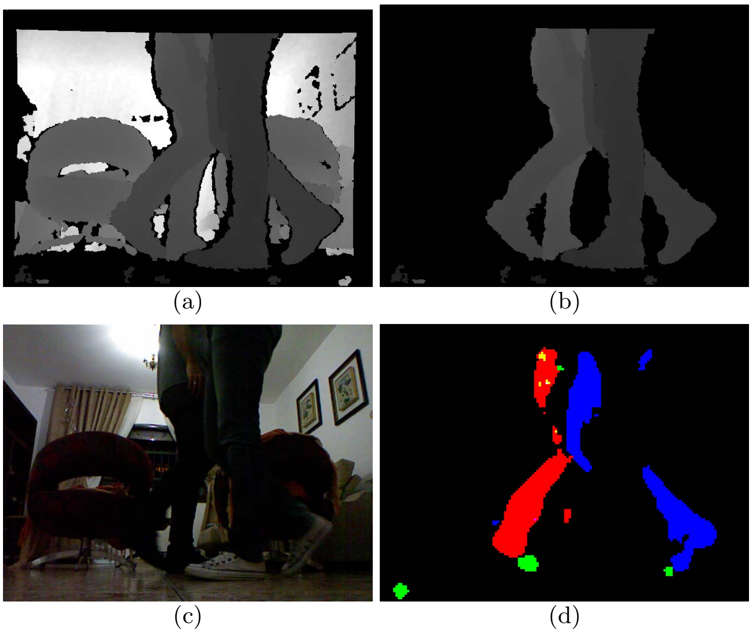

Fig. 3. Result obtained at a scene where two subjects are walking at opposite directions. This also exemplifies a case of occlusion between moving objects. **(a)** Depth image. **(b)** The same depth at image (a) after filtering for background elimination. **(c)** Texture image for the same scene shown at (a). **(d)** Detected motion represented as color labels.

The other subject, closer to the capturing device, executes a movement from right to left. Note the correct classification of both movements, even on the region where they intercept each other. The green pixels that arise on Figure 3(d) were identified as moving up, a reasonable result, except for the green blob that appears at left-bottom corner: it appears due to error on depth capturing. This same experiment is also an example of how our method gets successful results even in the presence of occlusion of moving objects. Note how a leg is partially occluded by another and still has its motion correctly identified.

Figure 4 shows two RGB-D frames on which a couple of dancers executes a movement to right with a subtle motion of the arms to up. The images at the bottom of the figure show the output obtained for patches of dimension 4×6 when the parameter α takes the values 0.00, 0.25, 0.50, 0.75 and 1.00. The best result was obtained for $\alpha = 0.50$. Black pixels indicate absence of motion.

For the input shown at left of Figure 5, we obtained the output shown at right of the same figure. This experiment shows the nice visual appealing of the matchings provided by the arrows. Instead of color labels, arrows with circled tips are used to indicate the motion.

Various values of the parameter α together with different patches sizes have been evaluated. As expected, our motion estimation achieved better results with

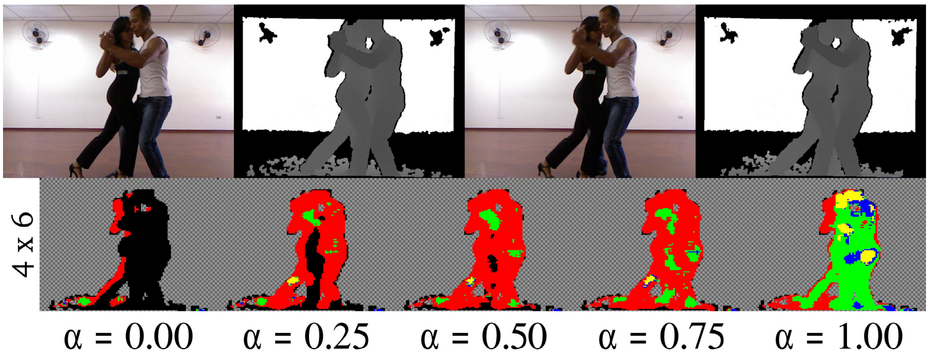

Fig. 4. Varying the values of parameter α

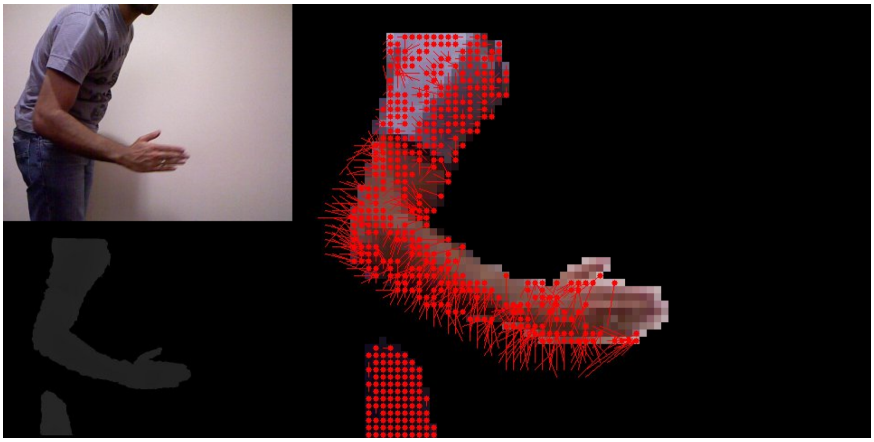

Fig. 5. Video sequence with a subject moving his arm. Matchings are shown as arrows, indicating that the arm is moving up and to the right direction.

intermediate values of α. For values close to the extremes of the valid range (0 and 1), the results were poor, indicating matchings that were not consistent with the observed movement.

4.1 Conclusions and Future Works

Our algorithm takes as input a sequence of pairs of registered RGB texture and depth map. Since the acquired input depth map is already registered to the texture image, there is no need for using knowledge about intrinsic or extrinsic calibration parameters between the infra-red light receptor and the camera.

The developed system has shown to be a convenient framework to deal with input data generated by the new depth-sensing devices. The application is capable of doing image processing and execute computer vision algorithms, thus allowing easy evaluation of results at real-time rates.

There are many features that may be considered to improve the results and the performance:

- Enforce coherence in the motion of nearby patches, which often present similar motions, turning the technique less sensitive to image noise and ambiguous results.
- The direction of the movement of the patches is calculated on 3D space but the visualization of the results is done only in the plane that is determined by texture coordinates. The development of a 3D visualization have already been started.
- It is possible to apply this method to detect the rigid parts of an articulated object.
- Compare our results with other 3D motion segmentation algorithms [3].

Our ongoing work include all these possibilities and new advances will be reported in due time.

References

1. Leyvand, T., Meekhof, C., Wei, Y.C., Sun, J., Guo, B.: Kinect Identity: Technology and experience. Computer 44, 94–96 (2011)
2. Gallo, L., Placitelli, A., Ciampi, M.: Controller-free exploration of medical image data: Experiencing the Kinect. In: CBMS, pp. 1–6 (June 2011)
3. Tron, R., Vidal, R.: A benchmark for the comparison of 3-D motion segmentation algorithms. In: IEEE CVPR, pp. 1–8 (2007)
4. Kumar, M.P., Torr, P.H.S., Zisserman, A.: Learning layered motion segmentations of video. In: ICCV (2005)
5. Suma, E.A., Lange, B., Rizzo, A.S., Krum, D.M., Bolas, M.: FAAST: The Flexible Action and Articulated Skeleton Toolkit. IEEE Virtual Reality, 245–246 (March 2011)
6. Noma, A., Graciano, A.B.V., Cesar Jr., R.M., Consularo, L.A., Bloch, I.: Interactive image segmentation by matching attributed relational graphs. Pattern Recognition 45(3), 1159–1179 (2012)
7. Noma, A., Velho, L., Cesar Jr, R.M.: A computer-assisted colorization approach based on efficient belief propagation and graph matching. In: Bayro-Corrochano, E., Eklundh, J.-O. (eds.) CIARP 2009. LNCS, vol. 5856, pp. 345–352. Springer, Heidelberg (2009)
8. Chui, H., Rangarajan, A.: A new point matching algorithm for non-rigid registration. Comput. Vis. Image Underst. 89(2-3), 114–141 (2003)
9. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Intl. Journal of Pattern Recognition and Artificial Intelligence 18(3), 265–298 (2004)
10. Cesar Jr, R.M., Bengoetxea, E., Bloch, I., Larranaga, P.: Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms. Pattern Recognition 38(11), 2099–2113 (2005)