Motion segmentation from texture and depth images using graph homomorphism

David da Silva Pires, Roberto M. Cesar-Jr
Department of Computer Science
Institute of Mathematics and Statistics (IME-USP)
São Paulo, Brazil
www.ime.usp.br/∼{cesar, davidsp}

Luiz Velho
Visgraf Laboratory
National Institute for Pure and Applied Mathematics (IMPA)
Rio de Janeiro, Brazil
http://lvelho.impa.br

Fig. 1. Detection and segmentation of movement (right) using our method applied to texture (left) and depth (middle) images. Red and blue colors indicate movement to right and to left, respectively.

***Abstract*—We present an approach for motion segmentation 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.**

I. Introduction

This work aims at showing the benefits of using the additional information given by the depth image registered with a texture image, presenting an algorithm that detects the direction of movement at real-time rates.

Given a video sequence, such as the one in Fig. 2, the application builds a graph for each frame and compare them.

Fig. 2. Input data is a video sequence of one texture and one depth map images per frame. The texture image provides RGB values while the depth map image gives us triplets (x, y, z).

We use the theory of 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 [1] and object tracking [2], among others [3].

II. Methodology

a) Input data: Our algorithm takes as input a sequence of pairs of registered RGB texture and depth map, which are captured from a Kinect™ device. The developed software also accepts input from a variety of other sources, such as video files, web-cam, video 4D files [4] and sequences of image files.

b) Texture and depth filters: Our system implements two texture filters: (i) flip on the horizontal direction and (ii) decreasing of the resolution. The first one is used because when a user interacts with the system, it is easier to look at the results as if it was in front of a mirror. The filter to decrease resolution, besides accelerating all the computations that are done, is closely related to two other factors: the velocity and the distance of each moving object to the capture device. We also implemented a depth map filter that allows us to choose data that is enclosed by two thresholds, determining the near and far planes of the visualization volume.

c) Graph based approach: In order to find a matching, we used six values for each pixel on input data: RGB and (x, y, z) data. The frame representation is given by an attributed relational graph. The recognition of the direction of the movement is done through an inexact graph matching.

d) Graph generation: The model and the input graph are built from consecutive pairs of texture and depth map input frames. In order to build the graph, we consider the representation of the input images (texture and depth map) composed by patches of $n \times n$ pixels.

e) Graph matching: A matching is an ordered pair of vertex descriptors, from the model to the input graph. For each vertex belonging to the model graph, we find the vertex on the input graph that is closest to it according to a distance

measured by a cost function. The cost function c is given by a weighted average between two distances: d_{RGB} and d_{XYZ}:

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

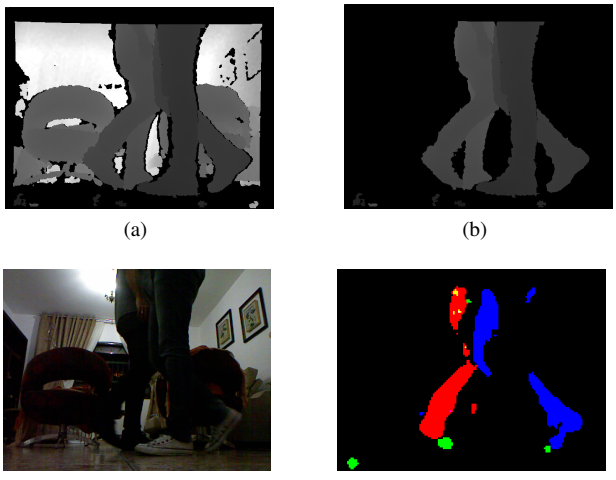
The d_{RGB} value measures the distance of the color of the patches being compared on RGB space, while the d_{XYZ} value measures the distance between the (x, y) texture coordinates and between the Z depth values. We used $\alpha = 0.5$ and implemented two distance functions: city block (Manhattan) and Euclidean.

Matching visualization: Matchings can be drawn in three ways: arrows, colored or gray-level patches. The first one draws, for each matching, an arrow that goes from a vertex location on the model graph to the position of the matching vertex on the input graph. The other modes represent each direction by a colored label and the program paints the whole area of each patch with the corresponding label.

Computation of directions: In order to compute the direction to which a patch is moving on, we calculate the arc-cosine of the normalized dot product between the vector that represents the shift of the patch and the unit vector $(1, 0)$, that points to the direction of the x axis. After that, we discover to which quadrant the vector belongs to by considering the signal of its y coordinate.

f) Median filter: Final results visualization shows that there are many patches that correctly identify the direction of the movement, but still others represent bad matchings. In order to eliminate these ones and treat this lack of spatial continuity, the median filter is applied to the image of the labeled patches.

III. Results and Discussion

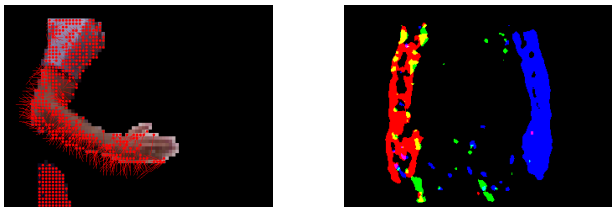


(a) (b) (c) (d)

Fig. 3. Typical input and output data.

Typical input and output data are shown in Fig. 3. Fig. 3 (a) shows the captured depth of two subjects walking at opposite directions, with occlusion occurring between them and also between their respective legs, while Fig. 3 (b) shows this depth after background elimination filtering. Fig. 3 (c) shows the texture for the same scene. Note that the depth is already registered with the texture. Finally, Fig. 3 (d) shows the detected motion represented as color labels. 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 capture device, as indicated by the gray levels in the depth map; it is walking from left to right. The other subject, closer to the capture 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 in Fig. 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 one and still has its motion correctly identified.

Fig. 4 (a) shows the nice visual appealing that the representation of matchings as arrows brings. Fig. 4 (b) exemplifies the fine grained result obtained when patches of size 2×2 pixels, instead of 10×10, are used in the video shown in Fig. 1.



(a) (b)

Fig. 4. Visualization of the matchings as arrows, indicating that the arm is moving up and to the right direction, and fine grained result using patches of 2×2 pixels.

IV. Future work

- To parallelize texture and depth filters to run on different processes, since they are independent from each other.
- To take a temporal approach to deal with patches that are in the interior of moving objects that are too thick and thus have the tendency to not be shown on results.
- To determine motion by considering directions on 3D space.
- To detect the rigid parts of an articulated object.

References

[1] A. Noma. (2012, May) Interactive natural image segmentation. [Online]. Available: http://structuralsegm.sourceforge.net/

[2] A. B. V. Graciano, "Rastreamento de objetos baseado em reconhecimento estrutural de padrões," Master's thesis, University of São Paulo, São Paulo, Brazil, March 2007.

[3] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Intl. Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 265-298, 2004.

[4] M. B. Vieira, L. Velho, A. Sá, and P. C. Carvalho, "A camera-projector system for real-time 3D video," in *Proceedings of IEEE International Workshop on Projector-Camera Systems (PROCAMS)*, San Diego, California, USA, June 2005, jointly with CVPR 2005.