Tabu search for human pose recognition

William Dyce, Nancy Rodriguez, Benoit Lange, Sebastien Andary, Antoine Seilles

Tabu search for human pose recognition

Dyce W.[a], Rodriguez N.[b], Lange B.[c], Andary S.[d], Seilles A.[d]

[a]Université Montpellier 2, 2 Place Eugène Bataillon, 34095 Montpellier Cedex 5, France;
[b]LIRMM, 161 Rue Ada 34090 Montpellier, France;
[c]UPMC Univ Paris 06, ICS - Institut du Calcul et de la Simulation, F-75005, Paris, France;
[d]NaturalPad, 70 Ancien Chemin de Saint Vincent Prades-le-Lez, 34730, France

ABSTRACT

The use of computer vision techniques to build hands-free input devices has long been a topic of interest to researchers in the field of natural interaction. In recent years Microsoft's Kinect has brought these technologies to the layman, but the most commonly used libraries for Kinect human pose recognition are closed-source. There is not yet an accepted, effective open-source alternative upon which highly specific applications can be based. We propose a novel technique for extracting the appendage configurations of users from the Kinect camera's depth feed, based on stochastic local search techniques rather than per-pixel classification.

Keywords: computer-vision, human-computer interaction, topology

1. INTRODUCTION

Computer vision techniques are widely used to extract meaningful human poses from a stream of images. Over the last few years Microsoft's Kinect depth sensor and accompanying software libraries (Kinect for Windows,[1] NITE,[2] ...) have lowered the barrier of entry to human movement recognition. Our interest is in using the Kinect for movement analysis, in line with the work of Oulasvirta et al.[3] and Stone et al.[4] We are interested in particular in applying this movement analysis in a therapeutic context, for automatic post-stroke mobility evaluation for example.

Microsoft's "Kinect for Windows" library estimates the user's pose in the form of tree graph of labeled nodes representing joints, and presents this to the application programmer via a software interface. Primesense, the company responsible for developing the Kinect sensor, has made available similar pose-tracking middleware. Both these libraries are closed-source[5] and implementation details are scant, but it is known that Microsoft Research reformulated the problem of body part recognition as a classification problem and used extensive machine learning to train their classifiers.[6]

Further author information:
Dyce W.: E-mail: william.dyce@univ-montp2.fr
Rodriguez N. E-mail: nancy.rodriguez@lirmm.fr
Lange B.: E-mail: benoit.lange@lip6.fr
Andary S.: E-mail: sebastien@naturalpad.fr
Seilles A.: E-mail: antoine@naturalpad.fr

There are many problems with this approach which hamper its use for movement analysis, as Oulasvirta et al. noted in the context of their work on body movements.[3] To begin with, poses not present in the training set (for instance lying down or facing away from the camera) are not detected. Furthermore since the algorithm simply connects the patches together to form a "skeleton", it is possible for the resulting skeleton to defy the constraints of the human body, for instance by putting an arm through its torso. There is also no "bone" rotation information: bone roll values are needed, for example, to measure "dorsiflexion" during a post-stroke Fugl-Meyer et al.[7] Assessment of Sensorimotor Recovery.

It is with these limitations in mind that we present the beginnings of an alternative solution to the real-time human pose estimation, or "user skeleton extraction", problem.

2. RELATED WORK

We look at the problem of user skeleton extraction as a general topology extraction problem. This subject has been studied for several years now within the 3D graphics community but it remains an open problem.

Baran et al.[8] create a signed distance field by sampling the model's mesh, then use its gradient to approximate a medial surface. Finally the model is packed with spheres along this surface: their intersections approximate the topology of the model. Au et al.[9] contract their model mesh down to a degenerate tree-shape using Laplacian smoothing, then collapse this into a curve skeleton. Other techniques use mesh sequences or transformations: Schaefer et al.[10] cluster groups of contiguous faces together when they can be described approximately by a rigid transformation, in other words a combination of translations and rotations. Once the faces have been clustered the underlying topological structure, the skeleton, can be calculated. Similarly de Aguiar et al.[11] use spectral clustering to associate vertices across full mesh sequences based on spatial affinity over time. These mesh-based techniques are unfortunately not well adapted to the depth maps provided by the Kinect, more readily converted into points clouds or voxels.

Kar et al.[12] use multiple Kinect sensors to build a closed voxel model of the user from which "voxel scooping"[13] extracts a skeleton to be trimmed and matched to an anatomical model of a human. Our current work is loosely inspired by that of Au, et al.[9] and Tierny, et al.[14] who use Morse Theory to deduce topology of a 3D mesh from the minimum and maximum values of a key function defined across its surface.

3. OUR APPROACH

For our implementation we used the open source OpenNI[15] library to stream depth-labeled pixels from the infrared sensor. OpenNI depth images (see figure 1) use lower values to represent closer areas and higher values to represent more distant ones, though the 0 value (black) is used for "dead" areas which are either too near, too far or subject to UV interference (from the sun or other Kinect sensors[16]).

Our method uses the entire set of depth-labeled pixels provided by the Kinect sensor. An initial preprocessing phase separates the user's silhouette from the background (see section 3.1). Next we approximate the appendage positions by calculating 5 paths from the centre of the silhouette with geodesically distant endpoints (see section 3.2). Finally we remove as many path nodes as is possible while keeping it within the bounds of the user silhouette (see section 3.3).

Figure 1. OpenNI source depth-map (normalised).

Figure 2. Foreground (user) segmentation.

3.1 User segmentation

This is achieved by calculating the histogram of the map's non-black areas and looking for the first "significant" peak. Let:

- $\tau \in [0; 640 \times 480]$
  the minimum surface in pixels of the user,
- $\alpha \in]0; 1]$
  the fraction of τ to used as a threshold for the nearest parts of the user,
- $\beta \in]0; 1]$
  the fraction of τ to used as a threshold for the furthest part of the user.

We scan the histogram's categories from nearest (darkest) to furthest (lightest) looking for one containing at least τ depth pixel elements. We then move backwards to the first category containing fewer than $\tau \times \alpha$ and forwards to the first category containing fewer than $\tau \times \beta$ elements. Everything outside of this interval of depth is set to 0 ("dead") on the map and ignored (see figure 2).

The cvBlob[17] library contains an implementation of the Senior et al.[18] blob detection algorithm. We use cvBlob to identify the pixel-area occupied by the user (assumed to be the largest blob) and filter out the rest. From here on our method uses the depth-labeled silhouette of the user.

3.2 User diameters

A graph of nodes is generated based on the remaining non-null pixels in the depth map. For each depth-pixel p in the user's silhouette we create a graph node n. Each neighbouring pixel p' of p will thus have a corresponding graph node n' if it is non-null. Where n' exists n and n' are connected if and only if the absolute difference in value (depth) $\delta(p, p')$ is below a given threshold ϵ. Thus if an arm is front of the body, for instance, paths will not cross it.

The weight of any connection (n, n') is equal to the sum of $\delta(p, p')$ and the Euclidean distance between p and p' on the depth map. This means that the differences in depth are taken into account when calculating the geodesic length of paths. It should be noted that for our prototype

we downsample the original image to $\frac{1}{8}^{th}$ its original size to speed up calculations: the performance increase more than makes up for the slight decrease in accuracy.

Tierny et al.[19] identify an initial set of prominent features based on the geodesic distances of each vertex from the extremities of the mesh diameter. Their algorithm makes no assumptions about the topology of the model, but since we are dealing with humans we can generally assume that there will be 5 appendages. As such we search not for the two most geodesically distant points, but rather for the 5 points most geodesically distant both from the blob centroid C and from each other.

These 5 optimal "appendage paths" are approximated sequentially using a modified version of Dijkstra's algorithm, rooted at the blob centroid. Let $\Delta(P)$ the length of a path P across the previously defined graph. We perform a best-first graph exploration in order to find P for which $\lambda(P) = \frac{\delta(P)}{tabu(P)}$ is maximal (see algorithm 1).

Algorithm 1 appendage(centroid)

```
frontier.push(centroid)
best ← centroid
while frontier is not empty do
    n ← frontier.pop()
    for n' in neighbours(n) do
        frontier.push(n')
        if (best.distance / best.tabu) < (n'.distance / n'.tabu) then
            best ← n'
        end if
    end for
end while
return best
```

"Tabu search" is a local optimisation technique proposed by Glover et al.[20] which avoids local optima by placing a tabou on previous results. Here we are maximising the length $\Delta(P)$ of each path while avoiding the same appendage being "traced" multiple times by different paths. The "tabu" factor $tabu(P)$ is a penalty applied to the utility of any path which revisits previously explored nodes of the depth map. This ensures that the 5 paths diverge. The penalty for traversing a given node is updated each time a new appendage is discovered: a negative utility value is propagated backwards from each new appendage end-point with exponential attenuation (see algorithm 2).

Algorithm 2 tabu(appendage, tabu)

```
frontier.push(appendage.end)
while frontier is not empty do
    n ← frontier.pop()
    for n' in neighbours(n) do
        frontier.push(n')
        tabu[n'.end] ← max(tabu[n'.end], 1/(length(n')^2))
    end for
end while
```

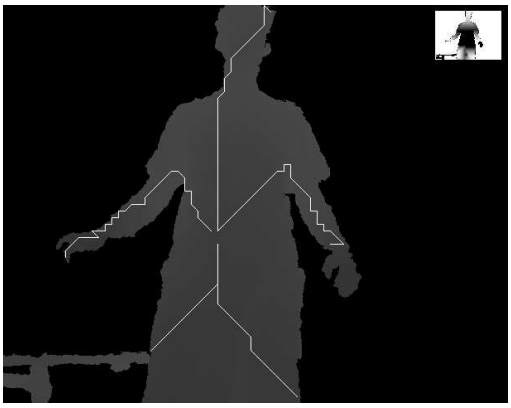

Figure 3. User appendage paths before simplification.

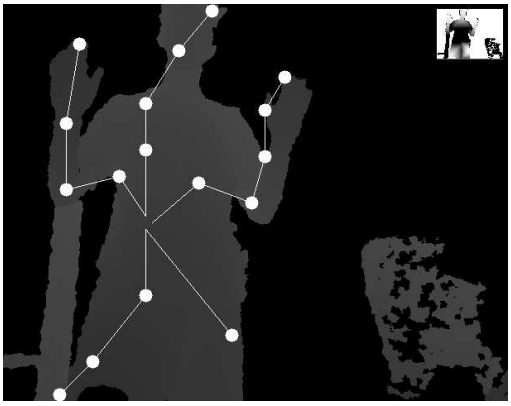

Figure 4. Final result after simplification.

3.3 Appendage simplification

We now have a set of 5 appendage paths rooted and the silhouette's centroid (see figure 3). Since the end goal is an articulated skeleton we need to remove as many nodes as possible from these paths so as to arrive at a small number of long, straight segments; However the resulting "bone" segments should not leave the user's silhouette (see figure 4).

Starting at the tip of the appendage path we explore backwards along its path towards the silhouette centroid, using Bresenham's line algorithm to perform "ray casts". In so doing we can calculate the number of pixels d outside of the silhouette that the candidate bone will cross. For b the length of the candidate bone and k a constant, a new bone is started when $b \times d > k$ and the process is repeated until we arrive back at the silhouette centroid.

4. RESULTS AND CONCLUSIONS

In this paper we provided a short summary of our technique for extracting the topology of a user from a single frame of depth-data in order to infer their current pose. This is done by finding 5 appendage paths with end-points that are geodesically distant both from the center of the user's silhouette and from each other.

Our application was developed in C++ using the OpenNI,[15] OpenCV[21] and cvBlob[17] libraries. As this is an exploratory study its main merit is the novelty of the technique employed; it is simply a proof of concept. We hope however with more work to attain real-time speeds.

Aside from speed optimisations, future work will look at ways to improve on the naïve foreground segmentation method, to place bone joints closer to the center of the blob using sphere-packing in a similar manner to Baran et al.[8] and to fit the final graph to an anatomical model of a human being for use in natural interaction applications. We hope for this to adapt the work of Kar et al.[12]

REFERENCES

[1] Microsoft, "Kinect for windows product page." http://www.microsoft.com/en-us/kinectforwindows/.

[2] Primesense, "Nite middleware." http://www.openni.org/files/nite/.

[3] Oulasvirta, A., Roos, T., Modig, A., and Leppänen, L., "Information capacity of full-body movements," in [*Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*], *CHI '13*, 1289–1298, ACM, New York, NY, USA (2013).

[4] Stone, E. and Skubic, M., "Evaluation of an inexpensive depth camera for in-home gait assessment," *J. Ambient Intell. Smart Environ.* **3**, 349–361 (Dec. 2011).

[5] Hendel, Z., "Openni and nite licenses." http://groups.google.com/forum/#!msg/openni-dev/kLwLLMmL5Bk/mAqupafe9TUJ.

[6] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A., "Real-time human pose recognition in parts from single depth images," in [*Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*], *CVPR '11*, 1297–1304, IEEE Computer Society, Washington, DC, USA (2011).

[7] Fugl-Meyer, A., Jääskö, L., Leyman, I., Olsson, S., and Steglind, S., "The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance.," *Scandinavian journal of rehabilitation medicine* **7**(1), 13–31 (1974).

[8] Baran, I. and Popović, J., "Automatic rigging and animation of 3d characters," *ACM Trans. Graph.* **26** (July 2007).

[9] Au, O. K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., and Lee, T.-Y., "Skeleton extraction by mesh contraction," in [*ACM SIGGRAPH 2008 papers*], *SIGGRAPH '08*, 44:1–44:10, ACM, New York, NY, USA (2008).

[10] Schaefer, S. and Yuksel, C., "Example-based skeleton extraction," in [*Proceedings of the fifth Eurographics symposium on Geometry processing*], *SGP '07*, 153–162, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2007).

[11] de Aguiar, E., Theobalt, C., Thrun, S., and Seidel, H.-P., "Automatic Conversion of Mesh Animations into Skeleton-based Animations," *Computer Graphics Forum (Proc. Eurographics EG'08)* **27**, xx–xx (4 2008).

[12] Kar, A., "Skeletal tracking using microsoft kinect," *Methodology* **1**, 1–11 (2010).

[13] Rodriguez, A., Ehlenberger, D. B., Hof, P. R., and Wearne, S. L., "Three-dimensional neuron tracing by voxel scooping," *Journal of neuroscience methods* **184**(1), 169 (2009).

[14] Tierny, J., Vandeborre, J.-P., and Daoudi, M., "Fast and precise kinematic skeleton extraction of 3d dynamic meshes," in [*Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*], 1–4 (2008).

[15] consortium, O., "Openni website." http://www.openni.org/.

[16] Butler, D. A., Izadi, S., Hilliges, O., Molyneaux, D., Hodges, S., and Kim, D., "Shake'n'sense: reducing interference for overlapping structured light depth cameras," in [*Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*], *CHI '12*, 1933–1936, ACM, New York, NY, USA (2012).

[17] nán, C. C. L., "cvblob." http://cvblob.googlecode.com.

[18] Senior, A., Hampapur, A., Tian, Y.-L., Brown, L., Pankanti, S., and Bolle, R., "Appearance models for occlusion handling," *Image and Vision Computing* **24**(11), 1233–1243 (2006).

[19] Tierny, J., Vandeborre, J.-P., Daoudi, M., et al., "3d mesh skeleton extraction using topological and geometrical analyses," in [*14th Pacific Conference on Computer Graphics and Applications (Pacific Graphics 2006)*], (2006).
[20] Glover, F., Laguna, M., et al., [*Tabu search*], vol. 22, Springer (1997).
[21] OpenCV, "Opencv website." `http://opencv.org/`.