Articulated Object Reconstruction and Motion Capture from Depth Video

Yuri Pekelny
Articulated Object Reconstruction and Motion Capture from Depth Video

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science

Yuri Pekelny

Submitted to the Senate of the Technion - Israel Institute of Technology

Adar 5767, Haifa
March 2008
This Research Thesis was done under the supervision of Prof. Craig Gotsman in the faculty of Computer Science.
LIST OF FIGURES

Figure 1: Rough user segmentation of the first frame of the “Tow Truck” sequence to bones ....... 9
Figure 2: Accumulated skin geometry. .................................................................................. 12
Figure 3: Skin geometry accumulation algorithm summary. ............................................... 12
Figure 4: Reconstruction algorithm summary. ................................................................. 17
Figure 5: Joint consistency improvement algorithm. ......................................................... 18
Figure 6: The size of accumulated skin geometry over time. ............................................. 23
Figure 7: ICP failure to register a bone, due to cylindrical symmetry of the bone. .............. 25
Figure 8: Incorrect segmentation....................................................................................... 26
Figure 9: Connecting the second joint of the occluded bone with two neighboring bones..... 33
Figure 10: The scanned models: Robot and Tow-Truck. ......................................................... 35
Figure 11: Scanning the dynamic Robot model (7 bones) - some of the input depth frames. .... 36
Figure 12: Scanning the dynamic Robot model (7 bones) - segmentation of the input frames into bones. ................................................................................................................................. 37
Figure 13: Scanning the dynamic Robot model (7 bones) - accumulated skin geometry in input poses. ................................................................................................................................. 38
Figure 14: Scanning the dynamic Robot model (7 bones) - complete model in input poses after skin reconstruction. ................................................................. 39
Figure 15: Scanning the dynamic Robot model (7 bones) - reconstructed skeletons. .............. 40
Figure 16: New poses of the robot model, created in Maya.................................................... 41
Figure 17: Scanning the dynamic Tow-Truck model (5 bones) - some of the input depth frames. 42
Figure 18: Scanning the dynamic Tow-Truck model (5 bones) - segmentation of the input frames into bones. ................................................................................................................................. 43
Figure 19: Scanning the dynamic Tow-Truck model (5 bones) - accumulated skin geometry in input poses. ................................................................................................................................. 44
Figure 20: Scanning the dynamic Tow-Truck model (5 bones) - complete model in input poses after skin reconstruction. ................................................................. 45
Figure 21: Scanning the dynamic Tow-Truck model (5 bones) - recovered skeletons. .............. 46
Figure 22: New poses of the tow truck model, created in Maya................................................ 47
Figure 23: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - frames from the synthetic animation sequence ................................................................................................................................. 48
Figure 24: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - input depth frames, as captured from the rendering z-buffer. ......................................................... 49
Figure 25: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - segmentation of the input frames into bones. ................................................................................................................................. 50
Figure 26: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - accumulated skin geometry in input poses. ................................................................................................................................. 51
Figure 27: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - complete model in input poses after skin reconstruction. ................................................................................................................................. 52
Figure 28: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - recovered skeletons. ................................................................................................................................. 53
LIST OF TABLES

Table 1: Algorithm runtime performance ................................................................. 20
Abstract

The aim of this work is to present an algorithm for acquiring the 3D surface geometry and motion of a dynamic piecewise-rigid object using a single depth video camera. The algorithm accumulates a separate point cloud for each rigid part of the scanned object. It begins from the point cloud of the first input frame and for each new frame adds points from the regions that were obscured to the camera in the previous frames. The more different regions are sampled from different viewpoints by the camera the more complete the reconstructed model becomes. In the end of the scanning process, each acquired point cloud is converted into a separate surface mesh. All rigid part meshes together construct the final 3D model.

The algorithm also reconstructs the dynamic skeleton of the object. The acquired model geometry and skeleton can be imported into a modeling system, in order to create new poses and animations of the reconstructed object.

During the reconstruction process the algorithm estimates the motion of each rigid part of the scanned object over frames, thus it can also be used for markerless motion capture. After the reconstruction process, the captured motion can be applied for the reconstructed 3D model and therefore can be observed from any viewpoint. The captured motion can also be retargeted to another 3D object skeleton, in order to animate a synthetic model based upon a motion which is captured from a real world object or actor.
## Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$</td>
<td>A point</td>
</tr>
<tr>
<td>nearest($p$)</td>
<td>A point in a point cloud, nearest to point $p$</td>
</tr>
<tr>
<td>$Z_{x}, Z_{y}$</td>
<td>Depth image gradients in $x$ and $y$ directions</td>
</tr>
<tr>
<td>$t$</td>
<td>Depth image index in the depth video sequence</td>
</tr>
<tr>
<td>$T$</td>
<td>The number of depth images in the depth video sequence</td>
</tr>
<tr>
<td>$u$</td>
<td>Joint coordinates</td>
</tr>
<tr>
<td>$T_{1}(u), T_{2}(u)$</td>
<td>Joint coordinates after applying the transformation $T_{1}/T_{2}$</td>
</tr>
<tr>
<td>$T_{b}^{t_{1}\rightarrow t_{2}}$</td>
<td>Transformation of bone $b$ from depth image $t_{1}$ to depth image $t_{2}$</td>
</tr>
<tr>
<td>$c$</td>
<td>Centroid of a set of points in the vicinity of the joint</td>
</tr>
<tr>
<td>$b, b_{1}, b_{2}$</td>
<td>Bone index</td>
</tr>
<tr>
<td>$B$</td>
<td>The number of bones in the subject</td>
</tr>
<tr>
<td>$d_{b}(p)$</td>
<td>The distance from point $p$ to bone $b$</td>
</tr>
<tr>
<td>$d(p)$</td>
<td>The euclidian distance from $p$ to nearest($p$)</td>
</tr>
<tr>
<td>$d_{t}(p)$</td>
<td>The distance from $p$ to the tangent plane at nearest($p$)</td>
</tr>
<tr>
<td>$d_{n}(p)$</td>
<td>The distance from $p$ to nearest($p$) projected onto the tangent plane at nearest($p$)</td>
</tr>
<tr>
<td>$C_{b}(p)$</td>
<td>A confidence measure for assigning a bone index $b$ to the point $p$</td>
</tr>
</tbody>
</table>
1. Introduction

Traditional 3D scanning applications involve only static subjects, therefore the main challenge is to produce an accurate digital model of the scene geometry. Over the past decade, a multitude of algorithms have been proposed to address this problem, and by now it may be considered (almost) solved. Thus, attention has shifted towards dealing with dynamic scenes, i.e. subjects that move.

Since the scene is dynamic, at first glance it may seem that the problem is not well-defined. What does scanning a scene, in which the geometry is constantly changing mean? What do we expect as the output of this process? The problem is compounded by the fact that in order to capture any motion accurately, we must sense the scene at real-time rates, a technological challenge for the scanning device in its own right.

In order to address the last challenge first, it seems that the most suitable device to use for dynamic scenes is the so-called depth video camera. Such a camera provides an image of the scene, where each pixel contains not only traditional intensity information, but also the geometric distance from the camera to the subject at that pixel. A number of commercial cameras generating this information at video rates have appeared over recent years [CVCM, 3DV, PS, VZS], and the state-of-the-art of the technologies involved is improving rapidly. Prices are also dropping, so we expect that depth video cameras will be available at reasonable cost within the next few years.

The simplest version of the dynamic scene scanning problem is the motion capture of a piecewise-rigid 3D subject (such as a person). This means that as output we are not interested in the precise geometry of the subject, rather in the rough motion of a “skeleton” representing its rigid parts, which are usually just a few. Motion capture is performed today using elaborate rigs involving markers placed on the subject, and it would be useful to have a device capable of markerless motion capture based only on depth cameras. This is the aim of a number of commercial companies [3DV, PS] who are developing depth cameras for use as motion capture and gesture recognition devices in interactive consumer-level gaming applications.
A more challenging version of the problem is full 3D scanning of dynamic piecewise-rigid 3D objects. The desired output is not just a skeletal model of the subject, rather a complete 3D model of the surface of each of the rigid components, and a description of the motion of each component over time. It would be ideal if all this could be computed in real-time, but even offline computation would be useful for many applications, especially those in which it is impossible to prevent the subject from moving during the scanning process.

The most challenging version of the problem is when the objects in the scene are deforming over time in a non-rigid manner. Here the description of the object motion is more complicated, and may not be parametric.

This work addresses the second question: scanning a piecewise-rigid subject. The goal of the algorithm is to recover the skeletal motion of the subject and to reconstruct its surface geometry from a sequence of range images captured by a single depth camera. The main issues are identification and tracking of the rigid components and reconstructing their surfaces despite partial occlusions present in each frame.

Since the system is markerless, it has no easily available correspondence information between points in different frames, thus it does not know which rigid component each sampled point (pixel) belongs to, nor does it know the rigid movement of any component between frames or the relative (rigid) camera transformation between frames. However, the temporal sampling rate is assumed to be sufficiently high, so that the relative position of each rigid component between each two consecutive frames is close enough to provide a good initial estimate for the classical Iterated Closest Point (ICP) registration algorithm.
2. Related Work

The problem of 3D model reconstruction of a rigid object has several solutions. Usually the idea is to use an algorithm that registers successive pairs of input frames, like the well known ICP method [BM92, CM92]. However, the pairwise approach always suffers from accumulation of error. Therefore to improve the final quality of the reconstructed model a global refinement process is applied [PLH02, P99]. This process minimizes the global registration error and combines all input data into one consistent representation.

For example, Rusinkiewicz et al [RHHL02] describe a real-time system for 3D model acquisition. This is an interactive system based on a structured-light depth scanner. It allows the user to move a scanned object in front of the 3D camera and to see the merged model on a computer screen immediately. Thus the user can decide how to move the object in real-time in order to acquire a complete 3D model.

Currently, motion capture methods used in the computer animation industry are marker-based, where the physical markers are placed on the moving subject. See, for example, [MG01] for a survey of human motion capture methods.

There are several algorithms in the literature of markerless motion which is captured from traditional video. Many of them [MCA06, CBK03, TAM*04, WL01] recover articulated object motion from visual hull sequences, acquired from object silhouettes. These algorithms require multiple synchronized video cameras observing the subject from several different directions. A very detailed description of markerless motion capture algorithm based on multiple view cameras is described in [K05].

There are also several model-based approaches [GWK05, RAPK06, BKM*04] capable of recovering articulated object pose from a single depth image by fitting it to a model. This, however, requires a very detailed and calibrated model in order that a single depth image suffice.

There are motion capture methods that combine depth and intensity data obtained from a depth camera. For instance, [CRHD00] improves articulated object poses estimation from
video data by utilizing depth camera information. This work assumes that the initial pose is given and uses it for partial assignment of input data pixel to rigid limbs. The process continues by alternately estimating each new frame pose with twist mathematics and estimating and improving limbs assignment of data pixels. Some of the ideas of that work are similar to the ideas of our work, however the final purposes are different.

The algorithm proposed in [AKP*04] recovers a full 3D model of the subject in addition to pose detection. This algorithm also does not require markers, but each input pose must be a complete surface mesh. Hence this algorithm is inapplicable to depth video input.

As opposed to simple motion capture, 3D model reconstruction of a non-rigid object is a more difficult problem which has not been studied much. Our aim here is to reconstruct a full 3D model of the subject. To register multiple components of an articulated subject, we assume a piecewise-rigid skeletal structure. A different approach was proposed in [MFO*07]. This algorithm assumes dense spatial and temporal sampling and performs the registration using kinematic properties of the space-time surface. The basic algorithm, which is designed for rigid body reconstruction, is extended to articulated or fully deformable objects by splitting the input data into small almost-rigid pieces.

Several authors have proposed algorithms which recover both the subject geometry and its motion. For example [AFW*99] proposes an algorithm for articulated model reconstruction. However the algorithm uses a pairwise frame registration approach, hence becomes very inefficient when applied to a long input data sequence. The authors also admit that the pairwise registration, as opposed to a more global approach, leads to suboptimal results.

Another recent work [WJH*07] addresses a similar problem, but with a deformable subject. It performs iterative reconstruction and optimization of the 4D structure assuming spatio-temporal smoothness. However the algorithm is computationally heavy and it is not clear if it is capable of reconstructing a full 3D model of a deformable object from depth video.

As far as we know, our method is the first one capable of reconstructing a complete 3D model of an articulate object using a single depth video camera. This is also the first markerless motion capture algorithm that does not require a template mesh of the subject.
3. The Algorithm

3.1. Assumptions and Outline

An articulated object consists of a set of connected rigid components. It is customary to refer to the rigid components as “bones”, and to the connection between two bones as a “joint”. The actual 3D geometry of the surface associated with the bone is called a “skin”. The algorithm makes the following assumptions on the input:

1. The motion of the subject is piecewise-rigid.
2. The combinatorial structure of the skeleton of the subject is known, i.e. \( n \) - the number of bones in the subject - is given, and we are given a list of ID pairs of connected bones.
3. All bones are (at least partially) visible in the first frame, and a rough assignment (segmentation) of image points to bones is given for this frame (see Figure 1).
4. During acquisition, the subject moves in such a way that most of its surface points are visible to the camera at some point in time.

Figure 1: Rough user segmentation of the first frame of the “Tow Truck” sequence to bones with different colors. Blue in right image means undecided bone index.
Note that the information we assume about the skeleton is purely combinatorial as opposed to geometric, namely, we do not know anything about the length of bones or the positions of joints. The algorithm tracks each bone of the subject almost independently throughout the sequence. Each image point is assigned a “bone index” – an integer in the set \{0, ..., n\}. The index 0 indicates that the point’s bone index is undecided, i.e. we have not been able to determine which bone the image point belongs to with sufficient confidence. These undecided points are typically in the vicinity of joints (where more than one bone competes for that point). The bone indices actually define the skinning weights of the skin geometry, while the possible values of each bone transformation weight are either ‘0’ or ‘1’.

The first objective of the algorithm is to accumulate over time the skin geometry of each bone. The second objective is to track in some coordinate system the motion of each bone over time.

The algorithm runs in several independent stages on each depth image of the sequence. For each new image the algorithm finds the new pose of each rigid bone by registering the accumulated bone skin geometry to the new image points. The algorithm verifies correctness of the registration process result by finding the joints coordinates of the skeleton and checking that the skeleton remains connected after applying the new estimated transformations. After that, it segments the new image points into bones, namely assigns bone indices to all image points. This is done by comparing distances from each new image point to the nearest points from each bone skin. Once the new image points are grouped into bones, the algorithm selects a subset of the new points that cover previously not sampled regions and add them to the accumulated skins. The following sections describe each stage of the algorithm in detail and elaborate on the relations between them. Figure 4 summarizes the algorithm.

3.2. Depth Image Preprocessing

Several parts of our algorithm require normal vectors of the input data points. To compute the normals we construct a simple mesh from the depth image points (pixels) and calculate the normals at vertices by area-weighted average of the normals to neighboring faces. To construct a simple mesh we use the rectilinear structure of the image grid. We generate all vertical, horizontal and the shorter diagonal edges, while ignoring those whose length is
larger than four times the grid size. The reason for this threshold is that two points with such a distance between them may belong to the same surface only if this surface is grazed by the camera’s at 75 degrees. However our depth camera isn’t capable of capturing surfaces viewed at such a grazing angle. After generating the edges we form triangles. This process also filters out single points which are far from the scanned point cloud.

An alternative way to calculate normal vectors is by calculating depth image (Z) gradients in x and y directions (Z_x and Z_y). Then the normal vector is (Z_x, Z_y, -1). However this method needs special treatment for boundary and missing grid points. Special treatment is also required for the case when the depth image contains points sampled from two objects, partially occluded one far behind another. The area-weighted average method overcomes this special case by dropping long edges.

The constructed mesh is also used to determine the set of boundary points in each image. This information is later used for ICP outliers detection, as described in Section 3.4.

3.3. Accumulating Skin Geometry

To reconstruct the skin geometry of a bone, the algorithm combines the depth samples from different frames by registering them in the coordinate system and the subject pose of the first frame. This results in a separate set of points (and their normals) accumulated for each bone. The skin geometry may be then reconstructed from this data, bone by bone, by any point cloud meshing algorithm (e.g. [KBH06, ACS*07]). See .

To register some point \( p \) from frame \( t \) into the coordinate system of the first frame, we need to know \( p \)’s bone index and the rigid transformation of the bone between this and the first frame. At each stage of the algorithm, when we obtain this information for a point, it may contribute to the reconstructed 3D model by applying the transformations. To reduce redundancy, thus limit the number of points in the reconstruction without compromising the skin completeness, we adopt only new points which are not too close to the set of points accumulated so far. As each sample point supposedly originates at the subject surface, our measure of the proximity of a point \( p \) with existing points on the subject is \( d_r(p) \) - the distance from \( p \) to the nearest point in the dataset – nearest\( (p) \) – projected onto the tangent plane of nearest\( (p) \). We add a new point to the skin geometry only if this distance is larger than a
threshold - typically half the grid size. Lowering this threshold will increase the number of points contributing to the surface model, but also increase the algorithm runtime. The more frames sampled from different viewpoints, the more complete the reconstructed geometry becomes.

Figure 2: Accumulated skin geometry. 3D points are accumulated for each bone separately (right). Each of these is later meshed using a point cloud meshing algorithm.

In practice, assignment of image points to the individual skins is done only after all bones have been identified in the new image. The bones indices assignment process doesn’t assign a bone index for each point, as described in Section 3.6. The points which bone can’t be determined with high probability are left “undecided” and they are not added to the accumulated skin geometry. Figure 4 summarizes the skin geometry accumulation algorithm.

For each point $p$ of the new depth frame $t$ assigned to bone skin $b$ do:

1. Calculate $p$’s position in $1^{st}$ frame by applying $T_{b}^{t\rightarrow 1}$.
2. Find nearest($p$) – the nearest point to $T_{b}^{t\rightarrow 1}(p)$ in bone skin $b$ (accumulated in the $1^{st}$ frame coordinates).
3. Calculate the distance of the projection onto the tangent plane of nearest($p$):

\[
d_t(p) = \sqrt{\left\|\text{nearest}(p) - T_{b}^{t\rightarrow 1}(p)\right\|^2 - \left(\hat{n}_{\text{nearest}(p)} \cdot T_{b}^{t\rightarrow 1}(p)\right)^2}
\]

4. If $d_t(p) > 0.5 \times \text{grid size}$ add $T_{b}^{t\rightarrow 1}(p)$ to bone skin $b$.

Figure 3: Skin geometry accumulation algorithm summary.
3.4. Finding Bone Transformations

The algorithm processes each input frame separately in chronological order. Each frame is analyzed based on the results from the previous frames. During this process the algorithm maintains two global data structures. One is the accumulated 3D points of the skins (each bone is accumulated separately). The other is the set of all rigid bone transformations from the first frame to all other frames. From this set of transformations we can find the relative transformation of any bone between any two frames.

To find the relative rigid bone transformation for each bone to a new input frame $t$, the algorithm registers the skin of that bone, accumulated so far, to the points of the new frame using the ICP registration algorithm [BM92, CM92] (See Appendix 1 for more details). Note that the registration is done for the accumulated bones skin to the new image pose, and not in the opposite direction. The reason for this is that during the registration process the new image points are not segmented to the rigid bones yet. We use the bone’s transformation from the first frame (where the reconstructed skin is accumulated) to the frame $t-1$ as the initial transformation for the ICP process. If the sampling rate is dense enough this is an excellent choice. The bones are registered in a hierarchical order, recursing (with BFS) through the skeletal structure, starting from the root bone.

The ICP algorithm is an easy and efficient registration algorithm. However, it is sensitive to the initial relative position of the registered data sets. If the sampling rate is not dense enough the ICP can get stuck in a local minimum and provide an incorrect registration result. To provide a good initial transformation for ICP it is possible to use an automatic process described in [GMGP05]. In practice our input frames were acquired from continuous depth video sequence, thus were quite close and did not require any preprocess.

There are several cases where the ICP registration may fail. For example, when the bone is occluded in the new frame or when the overlap between the skin and the points in the new frame is small. Our algorithm treats both cases as occlusion and provides a number of methods of detecting and recovering from ICP failure. For each bone we start by trying to find the rigid transformation using ICP without any additional constraints. If we fail, we find the joints locations between the current bone and all other bones connected to it (see Section 3.5) and repeat the ICP process together with the joints constraint (see Appendix 2). If we fail
again, the bone is considered occluded and we approximate its transformation using skeleton structure constraints, as described in Section 3.7.

The ICP algorithm is known to be sensitive to outliers. To detect such outliers, we employ a number of heuristics, similar to [RHHL02]. Corresponding pairs of points (identified as such by the ICP nearest neighbor pairing) of the following types are deemed outliers, hence ignored.

1) Pairs whose distance from each other is larger than a threshold.
2) Pairs whose angle between normals is larger than a threshold.
3) Pairs whose target point lies on the input mesh boundary.

To make the ICP algorithm more robust and accurate, we use dynamic thresholds for items 1) and 2). The initial distance threshold is 15x the image grid size and it decreases to 3x that after 5 iterations. The initial angle threshold is 80°, decreasing to 20° after 4 iterations.

3.5. Treating Occlusions

As mentioned above, the algorithm marks a bone occluded if the ICP registration algorithm failed. The algorithm has several ways to detect when the registration fails:

1) The number of corresponding ICP point pairs falls below a threshold.
2) The ICP algorithm does not converge quickly.
3) The resulting transformation does not satisfy the constraint connecting this bone with its parent bone – the distance between current bone joint transformed with the current bone transformation and the same joint transformed with the connected bone transformation is larger than a threshold.

Even if the bone is occluded, we would like to find an approximate transformation that satisfies the skeleton joint constraints, to keep the reconstructed skins connected. We find an approximate transformation of the occluded bone based on the transformations of the neighboring bones and skeleton joints constraints:

1) If the occluded bone is connected to just one bone, apply this neighbor’s relative transformation to the occluded bone in the frame in which it is occluded.
2) If the occluded bone is connected to two bones, use the rigid transformation that best fits both joint’s movements and minimizes the relative rotation between the bones.
3) If the occluded bone is connected to more than two bones, approximate its transformation by finding the rigid transformation that best fits all joints movement. See Appendix 4 for more details.

3.6. Bone Assignment (Segmentation)

Once we have found the relative transformation of all bones between the first frame and a new frame \( t \), we need to determine bone indices for the points in the new frame. This assignment then implies for each point a set of rigid transformations from frame \( t \) to all previous frames \( 1..t-1 \).

If the point \( p \) belongs to the skin of a bone \( b \), then the rigid transformation of \( b \) should translate \( p \) to the neighborhood of \( b \)'s skin. To determine which bone \( p \) best belongs to, we apply all possible bone transformations to \( p \) and calculate the distance between the result and the nearest point of \( b \)'s skin:

\[
d_b(p) = (1-\alpha)d_n(p)+\alpha d(p) \quad \text{(typically } \alpha=0.1)\]

where \( d_n(p) \) is the distance from \( p \) to the tangent plane at nearest\((p)\) – the local approximation of the reconstructed surface, and \( d(p) \) is the distance from \( p \) to nearest\((p)\). The purpose of the second term is to reject far points that fall near the tangent plane.

To decide which bone the point \( p \) belongs to among all candidates, we use the following confidence measure:

\[
C_b(p) = \frac{1}{d_b(p)} / \sum_{b \in \text{bones}} \frac{1}{d_b(p)}
\]

(3.2)

Note that

\[
\sum_b C_b(p) = 1
\]

(3.3)

After calculating confidences for all possible bone assignments we find \( b_{\text{max}} \) – the bone index \( b \) which maximizes \( C_b(p) \). This is just the bone index which minimizes \( d_b(p) \). However, we rely on this only if its confidence is significantly larger than the confidences of the other bones, namely, if \( C_{b_{\text{max}}}(p) > \tau \) (typically \( \tau=0.8 \)) and \( d_{b_{\text{max}}}(p) < \) sampling density, then we
assign \( p \) to \( b_{\text{max}} \). Otherwise \( p \)'s bone index remains undecided (=0) and the point does not contribute to skin geometry and future ICP registrations.

An alternative method of calculating the distances \( d(p) \) is by transforming all accumulated bone skins points into the coordinate system of the new frame \( t \) and computing the distance from \( p \) to each bone skin. This method may provide a better understanding of the distance based decision method. However the former method is more efficient because it doesn’t require applying a transformation for all accumulated points.

As mentioned in the previous section, some bones may be occluded, thus their transformations are only an approximation based on skeleton connectivity constraints. To avoid accumulating error due to these approximations, we consider as candidates only bones that are not occluded.

### 3.7. Tracking the Skeleton

A joint point \( u \) is a point connecting two bones \( b_1 \) and \( b_2 \). Therefore transforming \( u \) from the first frame to any other frame \( t \) with both bone transformations \( T_1 \) and \( T_2 \) should give the same result: \( T_1(u) = T_2(u) \). We seek a joint \( u \) that satisfies this property for transformations to all frames \( 2..t \). Therefore, following [AKP*04], for each pair of connected bones \( b_1 \) and \( b_2 \) we solve the following minimization problem for \( u \) (see Appendix 3 for more details):

\[
   u = \arg \min \sum_{i=1}^{t} \left\| T_{b_1}^i (u) - T_{b_2}^i (u) \right\|^2 + \gamma \| u - c \|^2
\]

(3.4)

Where \( T_{b}^i \) is the transformation of bone \( b \) in frame \( i \). \( c \) is the centroid of a set of points in the vicinity of the joint, namely a small set of points of \( b_1 \) close to \( b_2 \) and vice versa, in the first frame. Typically \( \gamma \approx 0.1 \). The second term stabilizes the solution in the cases where the first term admits more than one possible solution. As before, we ignore all transformation pairs that contain approximate transformations for occluded bones.

To complete the skeleton construction we choose the bone with the largest number of points in the first frame to be the root bone and connect all other bones in a hierarchical order.
3.8. Filter Skin Geometry

Bone skins are reconstructed by accumulating 3D points which are assigned to bones that provide the best registration for them (as described in Section 3.6). When two bones have similar transformations, it may be hard to determine which bone is a better match. An erroneous assignment may compound later in the process. To prevent this we try to identify points assigned incorrectly and remove them retroactively from the skins based on the information in each new frame.

First we apply the bones transformation from the first frame to the current frame for non-occluded bones. Then we check each pair of connected (and unoccluded) bones to see if they contain points representing two parallel surfaces. More precisely, we look for pairs of points \( p \) from \( b_1 \) and \( \text{nearest}(p) \) from the neighbor \( b_2 \), whose distance projected on the tangent plane of \( \text{nearest}(p) \) is shorter than the distance in the normal direction of \( \text{nearest}(p) \), i.e. \( d_t(p) < d_n(p) \). All points satisfying this condition are removed from the skin.

For \( t = 1..T \) do:
1. Calculate point normals and detect boundary points (Section 3.2)
2. Find a relative transformation for each bone in the reconstructed model to the new frame \( t \) (Section 3.4)
3. Based on the transformations and the new frame data, detect points with suspect bone assignments in the reconstructed model and remove them (Section 3.8)
4. Assign bone indices to points in the new frame \( t \) (Section 3.6)
5. Find joints locations basing on all bones transformations from frame 1 to frames 2..\( t \) (Section 3.7)
6. Add points from frame \( t \) to regions of the skins not yet covered (Section 3.3)

**Figure 4:** Reconstruction algorithm summary.

3.9. Joints Consistency Improvement

The rigid bones transformations and the skeleton joints locations are two outputs of the algorithm which depend upon each other. On the one hand, the joint locations are computed based on the recovered bones transformations (see Section 3.7). On the other hand, the bones transformations should be consistent with the joints. One possible solution to this chicken-
and-egg problem is an iterative solution where the joint locations in frame \( t \) are found based on the bone transformations of frames \( 1..t \) and used as a constraint for bone registration in frame \( t+1 \). However, from our experiments, this method does not give good results. The reason is that in order to find the precise location of joints we need to observe the bone transformations in a long sequence of input frames, and the results derived from a small set of initial input frames will be inexact. Using them as a constraint for further registration will lead to a suboptimal registration result, which will lead to even larger error in the joint locations. To prevent this error accumulation, our algorithm’s initial registration does not include the joints constraint. It uses the joint locations only to verify that the registration is consistent with the skeleton structure (see Section 3.4).

The final joint locations are found from bone transformations in all input frames. However the registration is never perfect. Hence applying the recovered transformations to the recovered skeleton separately for each bone may result in disconnected joints. These are not important if we are interested only in the reconstructed model. However if our objective is motion capture, disconnected joints are highly undesirable. To overcome this problem, after the final joint locations are found, the algorithm applies the algorithm described in Figure 5 for all input frames.

For all connected bone pairs \((b_{\text{parent}}, b)\) in hierarchical order do:

For \( t = 1..T \) do:

If both bones are not occluded in frame \( t \), and

\[
\left\| T_{b_{\text{parent}}}^t(u) - T_b^t(u) \right\| > 0.3 * \text{grid\_size},
\]

improve the \( T_b^t \) transformation by performing ICP registration for bone \( b \) with joint constraint \( T_{b_{\text{parent}}}^t(u) \).

Figure 5: Joint consistency improvement algorithm.
4. Experimental Results

4.1. Test Cases

The algorithm was applied to both synthetic and real-world depth video sequences. The calibration constants and thresholds used were the same for all input sequences.

Real world data was captured by a Vialux Z-Snapper depth camera (based on structured light technology) [VZS] for two physical models: a toy robot and a toy tow truck, consisting of 7 and 5 bones respectively (see Figure 10). Both motion sequences contain 90 frames, captured from a fixed viewpoint while the model was animated in front of the camera. The results are shown in the figures below. Figure 11 and Figure 17 contain a subset of the input depth video sequences. Since structured light scanning technology is based on illuminating the subject from an angle slightly different than that of the camera, there are occlusions due to “self-shadows” from the illumination angle, which can be quite significant at short ranges. Figure 12 and Figure 18 show the segmentation of the depth images to rigid components (“bones”) specified by the user in the first frame. Figure 13 and Figure 19 show all input preprocess meshes, separated to bones by the algorithm and accumulated in the same poses as captured in the sequence. Due to occlusions in the input, or undecided classification of image points, there are still surface regions where data is missing, resulting in some holes. Figure 14 and Figure 20 show the same poses after the bone geometries were meshed (as point clouds) to closed manifold surfaces (each bone separately) using the algorithm of [KBH06] (kindly provided by the authors). The results are quite good despite the noise and occlusions in the depth images. The reconstructed meshes together with the recovered skeletons where imported into Maya [AM] and new animations of the scanned objects were created. Figure 16 and Figure 22 show a number of new poses of the scanned objects.

While our algorithm is designed for subjects whose motion is piecewise-rigid, it is also applicable as an approximation of the motion of deformable subjects in some cases. Significant examples are models of skinned articulated bodies, such as human and animals. We tested this on a synthetic sequence (180 frames) of an articulated model of a Beast containing 12 bones. The motion sequences for this model were a variation on the motion
The inputs to our algorithm were the captured Z-buffers of the rendered animations.

The results on this synthetic deformable model are depicted in Figure 23 – Figure 28. As expected, since the motion is not truly piecewise-rigid, the results were not perfect. The segmentation procedure, which feeds the rest of the process, did not always classify the image points correctly. An example is the rear end of the subject (colored light blue in the figure) which was classified as part of the upper right leg instead of the torso (colored purple). This led to artifacts in the reconstruction, where that rear end sometimes protrudes, giving the impression of a “tail.”

4.2. Performance

The algorithm was implemented in MATLAB, and tested on a 2.8GHz Pentium D with 1GB of RAM. It was not seriously optimized for performance. With an efficient implementation and better hardware, we are confident that our algorithm could run in almost real-time, like [RHHL02]. The only part of the algorithm which has been optimized was finding nearest neighbors among points in 3D, since the algorithm spends a significant part of its runtime on these types of operations during ICP. We used the kd-tree based method implemented in the ANN C++ library [MA06], and yet still more that 60% of the runtime is spent on nearest neighbor searching. Table 1 gives the performance summary for three test sequences. The times quoted there do not include the pre-processed normal computations.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of bones</th>
<th>Number of frames</th>
<th>Average points per frame</th>
<th>Average time per frame (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truck</td>
<td>5</td>
<td>90</td>
<td>5,349</td>
<td>1.68</td>
</tr>
<tr>
<td>Robot</td>
<td>7</td>
<td>90</td>
<td>9,338</td>
<td>4.41</td>
</tr>
<tr>
<td>Beast</td>
<td>12</td>
<td>180</td>
<td>4,204</td>
<td>2.50</td>
</tr>
</tbody>
</table>

Table 1: Algorithm runtime performance

The algorithm runtime is affected by three main factors:
1. The number of depth points in each image: the size of the input frames affects the size of the accumulated skin geometry. The number of points in the accumulated skin geometry
affects the runtime of nearest neighbour searching in all three main steps of the algorithms: bone registration, bone indices assignment and skin accumulation.

2. The number of bones: the algorithm performs a registration step for each bone in each new input frame. The large number of bones requires more registrations, therefore slows down the algorithm. Another step which is affected by the number of bones is the bone index assignment. In this step each point from the new frame must be tested to determine whether it belongs to each of the model bones.

3. A less obvious factor is the number of bone occlusions during the depth video sequence. Each bone occlusion is detected after two registration attempts fail – one without any additional constraints and the other with the skeleton joints consistency constraint. Moreover, after a bone is marked as occluded, each successful registration of a neighboring bone results in additional registration attempts. Each such attempt usually requires more time than an average registration of an unoccluded bone because the registration error is large and the registration stops only after the maximum number of iterations is reached.

One of the objectives of the algorithm is markerless motion capture. Therefore the algorithm was designed to be capable of handling very long input sequences. The longest input sequence the algorithm was tested on was 1000 frames of animation of the synthetic beast model. The average runtime for this input was 3.739 seconds per frame.
5. Complexity Analysis

5.1. Space Complexity

The algorithm performs a single pass over the input depth video sequence. It does not need to store all the information from past depth frames. The only two global data structures the algorithm maintains are the accumulated skin geometry of the model and the set of rigid bones transformations. The rigid bones transformations set contains a 4x4 matrix for each bone in each input frame. The accumulated skin geometry is a set of points registered to the coordinates and model position of the first input frame. For each accumulated point the algorithm stores its 3D position, normal and bone index. The algorithm filters out points that do not contribute new geometric information to the reconstructed model, as described in Section 3.3. However, in the worst case, when all input points sample different regions of the model, the accumulated skin geometry may contain all input points. Therefore the worst case space complexity is:

\[ O(T \times P + T \times B) \]  

(5.1)

Where \( T \) is the number of depth images in the depth video sequence, \( P \) is the average number of points in each depth image and \( B \) is the number of bones in the subject.

In our test cases, after the model was sampled from different points of view, the growth of the accumulated geometry dropped dramatically. Figure 6 shows the growth of accumulated model size over a sequence of 600 frames of our beast model. In this example the final number of accumulated points was less than 2% of the total number of input points.
Figure 6: The size of accumulated skin geometry over time.

5.2. Time Complexity

The algorithm performs the following set of operations for each input frame:

1. Registration of accumulated model points to the new depth image
2. Segmentation of the new depth image to rigid components (bones)
3. Accumulation of new points
4. Accumulated model filtering
5. Skeleton joints locations extraction

The ICP algorithm finds the nearest neighbour in the target point set for each point from the
registered data set in each iteration. The complexity of nearest neighbour searching in a set of
$N$ target points using kd-tree is $O(\log N)$ for each point, plus $O(N\log N)$ for the kd-tree
construction. The number of iterations of our ICP registration process is bound by a constant.
Therefore the complexity of our ICP registration implementation for $N$ target points and $M$
source points is $O(N\log N + M\log N)$.

According to Section 5.2 the worst-case size of the accumulated model after $t$ frames is $P*^t$.
Step (1) requires nearest neighbour search from the accumulated point set to new depth image
points – $O(P\log P + t*P\log P)$. Steps (2)-(4) require nearest neighbour search in the opposite
direction – from the new depth image points to the set of accumulated points– $O(t*P\log t*P +
P\log t*P)$. Step (5) solves a least squares equation which is derived from all bones rigid
transformation acquired until current frame, therefore its complexity is $O(t*B)$. The total time complexity over $T$ frames is:

$$\sum_{t=2}^{T} \left[ O(P \log P + tP \log P) + O(tp \log tp + P \log tp) + O(tB) \right] = O(T^2P \log TP + T^2B) \quad (5.2)$$
6. Limitations

A key component of our algorithm is the ability to detect whether the inter-frame ICP registration succeeded or not. This detection prevents incorrect segmentation, which could be damaging. For some types of structured data, this detection mechanism fails. For instance, the model of a deforming human hand (see Figure 7). The cylindrical symmetry of the fingers confuses the ICP because there are many seemingly valid, but wrong, registrations. So even if the camera rotates 360° around a finger, the parts of the finger visible in each frame will always be registered to each other in one region of the finger, instead of completing the entire finger circumference. Consequently, we generate both incorrect motion capture and an incorrect 3D skin.

![Figure 7: ICP failure to register a bone, due to cylindrical symmetry of the bone. The red skin remains incomplete.](image)

A possible solution to this problem is to supplement the registration algorithm by using also the camera intensity channel as a 4th data dimension. Another solution could be replacing the ICP registration with any other algorithm capable of registering two partial point clouds of a rigid body.
Another problem we encountered is the problem of incorrect segmentation in the case of two “sliding” parallel planes. For example, the left leg of our robot model (colored in light blue in Figure 8) can only move parallel to the left side of the robot’s body (colored in green). In the region marked in Figure 8, the upper light blue points were mistakenly assigned to the leg and the lower green points were mistakenly assigned to the body. The reason for this is that when we have two small transformations perpendicular to the plane normal, both transformations align the points of one plane very close to other points of that plane. If one of these transformations brings the point to an unscanned region of that plane, the algorithm will decide that the other transformation is the best one for this point, which can be incorrect.

![Figure 8: Incorrect segmentation](image)

*Figure 8: Incorrect segmentation (of the light blue and green points inside the black circle) due to “sliding planes”.*
7. Summary and Conclusion

We have presented an algorithm capable of reconstructing a full 3D model of a dynamic articulated subject from a sequence of depth images taken by a single depth video camera. The algorithm segments the input set of point clouds to rigid components, detects rigid motions between the frames and reconstructs an articulated 3D model in a single pass over the data. Another pass may be performed in order to improve the joints consistency during motion. The algorithm detects bone occlusion and approximates skeleton-consistent motion for these occluded bones. In addition to 3D scanning, our algorithm tracks the skeleton of the subject, thus can be used for markerless motion capture over long sequences.

Future versions of this work will improve the approximation in the case of a deforming subject which is not truly piecewise-rigid. In the performance domain, our implementation currently runs offline after the data has been captured. The holy grail is, of course, to optimize the performance of the system so that it may run in real-time. This will enable a variety of important applications in the entertainment world.
8. Future Work

As mentioned above, our algorithm was implemented in MATLAB and was not optimized for performance. An efficient C++ implementation and algorithm optimization could result in a significant speedup in runtime.

Another open issue is the usage of intensity information, which is also available from the depth camera and currently ignored. This information may contribute to the registration step and help overcome some major algorithm limitations (see Section 0). The intensity information can also be used for texture mapping of the reconstructed 3D model.

The current algorithm requires manual bone segmentation in the first input frame. It would be nice to develop an algorithm that would be able to provide automatic bone detection in range video data. It does not seem that this can be achieved from a single depth image. But perhaps a depth video sequence where each rigid part moves in a slight different manner would help. A possible approach would be to register the first frame to the others with a non-rigid registration like [AKP*04], and then segment it to rigid parts with an algorithm like the one used in [SY07].

Another input currently provided by user is the set of pairs of connected bones. These pairs provide a combinatorial skeleton structure and indicate which pairs of bones should be used to find joint locations. An automatic identification of connected bones is another challenge.

Despite the fact that the algorithm assumes a piecewise rigid model, it can be used for a skinned model like in the synthetic beast example. However, the reconstructed 3D model will always consist of a set of pure rigid parts, which makes it not really useful for creating new poses and animation. A challenging issue is converting the accumulated point cloud to a skinned model. A single connected mesh of the complete model can be created by applying the point cloud meshing algorithm [KBH06, ACS*07] to all accumulated points together (rather than applying it to each bone separately). An open question is then how to assign bone skinning weights to each point of the reconstructed mesh such that the mesh will behave naturally when animated.
Appendix 1 : ICP Registration Algorithm

This section summarizes the methods used for registration step of the algorithm.

There are two basic methods for ICP (Iterative Closest Points) registration which were proposed independently in 1992. The first is known as point-to-point ICP, described by Besl and McKay [BM92]. The second is point-to-plane ICP, described by Chen and Medioni [CM92].

The two basic components of the ICP algorithm are: 1. Find corresponding pairs between two data sets and 2. Compute a registration error between the two. After that the algorithm finds a rigid transformation that brings one data set \( \{ p_i \} \) closer to the target set \( \{ q_i \} \) and thus reduce the registration error. This process is repeated iteratively until convergence. In each step the algorithm reduces the registration error, therefore it always converges to a local minimum. If initially the data sets are close enough, the algorithm usually converges to a global minimum, thus a good registration, in a small number of iterations.

The, point-to-point ICP algorithm proposes to minimize the average square distance between a set of corresponding pairs of points - \( \{ p_i \} \) and \( \{ q_i \} \):

\[
Err_i = \frac{1}{N_p} \sum_{i=1}^{N_p} \| q_i - R\tilde{p}_i - \tilde{t} \|^2
\]  

(1)

where \( R \) is a 3x3 rotation matrix and \( \tilde{t} \) is a translation vector.

\( R \) is parameterized using the following closed-form:

\[
R = \begin{bmatrix}
q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_3q_0) & 2(q_1q_3 - q_2q_0) \\
2(q_1q_2 - q_3q_0) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\
2(q_1q_3 + q_2q_0) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2
\end{bmatrix}
\]

(2)

Where \( [q_0, q_1, q_2, q_3] \) is the unit quaternian vector obtained from the unit eigenvector that corresponds to the maximum eigenvalue of the following symmetric matrix:

\[
Q(\sum_{pq}) = \begin{bmatrix}
tr(\sum_{pq}) & \Delta^T \\
\Delta & \sum_{pq} + \sum_{pq}^T - tr(\sum_{pq})I_3
\end{bmatrix}
\]

(3)

where \( \sum_{pq} \) is the cross covariance matrix, Eq. (4), \( \Delta = [A_{23} \quad A_{31} \quad A_{12}]^T \),

\( A_y = (\sum_{pq} - \sum_{pq}^T)_{ij} \) and \( \tilde{\mu}_p \) and \( \tilde{\mu}_q \) are the centers of mass of the points sets, as in Eq. (5).
\[ \sum_{pq} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left[ \tilde{p}_i \tilde{q}_i^T \right] - \tilde{\mu}_p \tilde{\mu}_q \]  
(4)

\[
\tilde{\mu}_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \tilde{p}_i 
\]

\[
\tilde{\mu}_q = \frac{1}{N_q} \sum_{i=1}^{N_q} \tilde{q}_i 
\]

The translation vector is obtained as:

\[
\tilde{t} = \tilde{\mu}_q - R \tilde{\mu}_p 
\]

The point-to-plane ICP algorithm uses the fact that the registered points sets actually represent two surfaces. Instead of building point-to-point constraints, this algorithm minimizes the average distance from the points set \( \{p_i\} \) to the local surface near \( \{q_i\} \), approximated by a tangent plane. The distance from point \( p \) to a plane passing through point \( q \) with a unit normal \( n \) is:

\[
\left\| n \circ (q - p) \right\| 
\]

Therefore the average distance error function becomes:

\[
Err_2 = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| \tilde{q}_i \circ (\tilde{q}_i - R(\tilde{p}_i) - \tilde{t}) \right\|^2 
\]

The advantage of the point-to-plane algorithm is faster convergence.

In order to add the joints constraints to the registration problem we would like to express the registration error function minimization solution as a set linear equations. To do this we use the linearization of exponential rotation coordinates [MLS94], also known as twist coordinates or instantaneous kinematics [PLH02], as follows:

The position of vertex \( v \) after a small rigid transformation can be approximated by:

\[
v' = v + r \times v + t 
\]

where \( t = (t_x, t_y, t_z) \) is a translation vector and \( r = (r_x, r_y, r_z) \) is a rotation vector.

The rotation vector \( r \) can be converted to a 3x3 rotation matrix \( R \) using Eq. (10).

\[
R = \exp(\hat{r}) 
\]

where \( \hat{r} \) is the skew-symmetric matrix in Eq. (11) and \( \exp(\hat{r}) \) is the matrix exponent in Eq. (12).
\[
\hat{r} = \begin{bmatrix}
0 & -r_z & r_y \\
r_z & 0 & -r_x \\
-r_y & r_x & 0 \\
\end{bmatrix}
\]  \hspace{1cm} (11)

\[
\exp(\hat{r}) = I + \hat{r} + \frac{\hat{r}^2}{2!} + \frac{\hat{r}^3}{3!} + \cdots
\]  \hspace{1cm} (12)

This approximation makes it possible to express the point-to-plane registration error function minimization as the following system of linear equations:

\[
\begin{bmatrix}
\vec{p}_1 \times \vec{n}_1, & \vec{n}_1 \\
\vec{p}_2 \times \vec{n}_2, & \vec{n}_2 \\
\vdots \\
\vec{p}_N \times \vec{n}_N, & \vec{n}_N
\end{bmatrix}
\begin{bmatrix}
\begin{pmatrix}
  r_x \\
r_y \\
r_z
\end{pmatrix}
\end{bmatrix}
=
\begin{bmatrix}
\begin{pmatrix}
  \vec{n}_1 \circ (\vec{q}_1 - \vec{p}_1) \\
  \vec{n}_2 \circ (\vec{q}_2 - \vec{p}_2) \\
  \vdots \\
  \vec{n}_N \circ (\vec{q}_N - \vec{p}_N)
\end{pmatrix}
\end{bmatrix}
\]  \hspace{1cm} (13)
Appendix 2 : ICP Registration Algorithm with Joint Constraints

As described in the Appendix 1, the rigid transformation can be expressed through a twist pair \((r, t)\) and the approximate location of a point \(v\) after a rigid transformation can be expressed by the linear formula in Eq. (9).

As described in Section 3.4, if the ICP registration without any additional constraints fails, the algorithm adds joint constraints to the problem. Each joint constraint is a coordinate \(u'\) where the current bone should be connected to a neighbour bone. The number of joint constraints is equal to the number of successfully registered neighbor bones. Each joint coordinate \(u'\) is a target coordinate for the joint coordinate in the previous frame \(v\) transferred with the registered bone transformation. Each joint coordinate constraint can be expressed as linear system in Eq. (14) or its vector form in Eq. (15).

\[
\begin{pmatrix}
0 & u_z & -u_y & 1 & 0 \\
-u_z & 0 & u_x & 0 & 1 \\
u_y & -u_x & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
r_x \\
r_y \\
r_z \\
t_x \\
t_y \\
t_z
\end{pmatrix}
= \begin{pmatrix}
u'_x - u_x \\
u'_y - u_y \\
u'_z - u_z
\end{pmatrix}
\tag{14}
\]

\[
\begin{pmatrix}
-\hat{u} \\
I_3
\end{pmatrix}
\begin{pmatrix}
r \\
t
\end{pmatrix} = \begin{pmatrix}
u' - \bar{u}
\end{pmatrix}
\tag{15}
\]

The \((r,t)\) solution of the joint-constrained ICP problem is obtained as the least squares solution of the overconstrained linear system in Eq. (16)

\[
\begin{pmatrix}
A \\
N_p \\
\frac{3}{3} \cdot A_u
\end{pmatrix}
\begin{pmatrix}
r \\
t
\end{pmatrix} = \begin{pmatrix}
b \\
b_u 
\end{pmatrix}
\tag{16}
\]

where \(A\) and \(b\) are the coefficients derived from the ICP constraints and \(A_u\) and \(b_u\) are the coefficients derived from the joint constraints.
Appendix 3 : Estimating Occluded Bone Transformations

When a bone is occluded the algorithm cannot find the exact transformation of the bone. Still, the skeleton structure should remain connected, thus the algorithm estimates the transformation from the constraints imposed by the unoccluded bones connected to the occluded one. We distinguish between three cases, depending on the number of available constraints:

1. The occluded bone is connected to a single bone: the algorithm applies the transformation of the connected bone to the occluded bone.

2. The bone is connected to two neighboring bones: first the algorithm applies to the occluded bone the transformation of the connected bone which is higher in the skeleton hierarchy (the parent bone). This guarantees that the occluded bone remains connected to the parent bone. After that the algorithm applies a rotation around the axis from Eq. (17), centered at the parent bone connection point \( p_1 \) and rotation angle from Eq. (18) (see Figure 9).

\[
\text{axis } = (p_2 - p_1) \times (p_3 - p_1) \tag{17}
\]

\[
\alpha = \cos \left( \frac{p_2 - p_1}{\|p_2 - p_1\|} \cdot \frac{p_3 - p_1}{\|p_3 - p_1\|} \right) \tag{18}
\]

3. The bone is connected to three or more neighboring bones: the algorithm finds the occluded bone transformation by applying the closed-form quaternion solution for finding a rigid transformation described in Appendix 1. \( \{p_i\} \) are the current joints locations of the occluded bone and \( \{q_i\} \) are the desired locations found by applying the relative transformations of the connected bones.

Figure 9: Connecting the second joint of the occluded bone with two neighboring bones
Appendix 4: Estimation of Joint Positions

In order to find the exact position of a joint $u$ which connects bones $b_1$ and $b_2$ in the pose of the first input frame we need to solve the following minimization problem:

$$
\vec{u} = \arg \min \sum_{i=1}^{t} \left\| T_{b_1}^i (\vec{u}) - T_{b_2}^i (\vec{u}) \right\|^2 + \gamma \left\| \vec{u} - \bar{c} \right\|^2
$$

(19)

Where $T_b^i$ is the transformation of bone $b$ from the first frame to frame $i$, $c$ is the centroid of a set of points in the vicinity of the joint in the first frame and $\gamma=0.1$.

Each rigid transformation $T_b^i$ can be expressed as a 3x3 rotation matrix $R_b^i$ and 3x1 translation vector $t_b^i$:

$$
T_b^i (\vec{u}) = R_b^i \vec{u} + t_b^i
$$

(20)

Thus the minimization problem becomes:

$$
\vec{u} = \arg \min \sum_{i=1}^{t} \left\| (R_{b_1}^i - R_{b_2}^i) \vec{u} + (\vec{t}_{b_1}^i - \vec{t}_{b_2}^i) \right\|^2 + \gamma \left\| \vec{u} - \bar{c} \right\|^2
$$

(21)

By differentiating the cost function we obtain the linear system from Eq. (22) or Eq. (23) which is solved using least squares.

$$
\sum_{i=1}^{t} (R_{b_1}^i - R_{b_2}^i)^T ((R_{b_1}^i - R_{b_2}^i) \vec{u} + (\vec{t}_{b_1}^i - \vec{t}_{b_2}^i)) + \gamma (\vec{u} - \bar{c}) = 0
$$

(22)

$$
\left( \sum_{i=1}^{t} (R_{b_1}^i - R_{b_2}^i)^T (R_{b_1}^i - R_{b_2}^i) + \gamma I_3 \right) \vec{u} = \gamma \bar{c} - \sum_{i=1}^{t} (R_{b_1}^i - R_{b_2}^i)^T (\vec{t}_{b_1}^i - \vec{t}_{b_2}^i)
$$

(23)
Figure 10: The scanned models: Robot and Tow-Truck.
Figure 11: Scanning the dynamic Robot model (7 bones) - some of the input depth frames. Note the different camera viewpoints and model poses.
Figure 12: Scanning the dynamic Robot model (7 bones) - segmentation of the input frames into bones. Dark blue regions are undecided.
Figure 13: Scanning the dynamic Robot model (7 bones) - accumulated skin geometry in input poses. Holes are present where the surface was occluded in all input frames.
Figure 14: Scanning the dynamic Robot model (7 bones) - complete model in input poses after skin reconstruction.
Figure 15: Scanning the dynamic Robot model (7 bones) - reconstructed skeletons.
Figure 16: New poses of the robot model, created in Maya.
Figure 17: Scanning the dynamic Tow-Truck model (5 bones) - some of the input depth frames. Note the different camera viewpoints and model poses.
Figure 18: Scanning the dynamic Tow-Truck model (5 bones) - segmentation of the input frames into bones. Dark blue regions are undecided.
Figure 19: Scanning the dynamic Tow-Truck model (5 bones) - accumulated skin geometry in input poses. Holes are present where the surface was occluded in all input frames.
Figure 20: Scanning the dynamic Tow-Truck model (5 bones) - complete model in input poses after skin reconstruction.
Figure 21: Scanning the dynamic Tow-Truck model (5 bones) - recovered skeletons.
Figure 22: New poses of the tow truck model, created in Maya.
Figure 23: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - frames from the synthetic animation sequence.
Figure 24: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - input depth frames, as captured from the rendering z-buffer.
Figure 25: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - segmentation of the input frames into bones. Dark blue regions are undecided. Note the incorrect classification of the rear end as part of the (light blue) right leg instead of the (purple) torso.
Figure 26: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - accumulated skin geometry in input poses. Holes are present where the surface was occluded in all input frames.
Figure 27: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - complete model in input poses after skin reconstruction.
Figure 28: “Scanning” the synthetic deformable Monster model, consisting of 12 “bones” - recovered skeletons.
Bibliography

[3DV] http://www.3dvsystems.com


[CMU] CMU Graphics Lab Motion Capture Database, http://mocap.cs.cmu.edu


המחקרים המשותפים קבעו שלמה תלת מימדי של גוף בין שני קשיחים העצומים [CM92, BM92, RHHL02]. יৎכן עצב חדש בין שני קשיחים העצומים באמצעות טכניקת הפגזות, אך ההחלפה של מסלול התודעה של גוף בין שני קשיחים העצומיםいっぱい מגדיר את מספר תתרחשות בשתי התודעות של גוף בין שני קשיחים העצומים [MFO*07, WJH*07].

בנוסף, נており את המחקרים של מורכבים עזים ובין שני קשיחים העצומים באמצעות טכניקת הפגזות [RHHL02]. מחקרים שונים הראו שה解放 של תלת מימדי העצומים באמצעות טכניקת הפגזות ירידה ביכולת מסלול התודעה של גוף בין שני קשיחים העצומים.

מתוך וידאו, מעוררת להצלחת שלם תלת מימדי העצומים מחקרי זה מציג אלגוריתם לבניית מודל בנויה מאוסף של תתי חלקים המצלמה על ידיה. האלגוריתם מניח של מצלמת עומק אחת מעבר בהעמם חלק קשיח של תהליך ענני נקודות נפרד לכל צובא אלגוריתם ה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל, החדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהজאז של המודל נשבר ותלוי ואת האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל, החדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים בנפרד. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים את הנקודות ענני נקודות המשוחזרים אלגוריתם מוסיף ל,חדשתמונה עבור כל הקלטים בווידאו. כל הנקודות מהאלגוריתם משוחזרים שהזאז של המודל נשבר ותלוי את האלגוריתם משוחזרים של המודל נפרדים ביחידה. האלגוריתם יכין בינה קשיחים בין קשיחים העצומים של מודל תלת מימדי העצומים כולל טכניקת הפגזות [MFO*07, WJH*07].
1. The algorithm builds on sections of rigid parts glued together, which are also pairs of stick parts. This information constitutes the combinatorial structure of the model. The algorithm also uses this information to find the exact structure of the rigid parts and their precise position in space.

2. A partial partition of the parts is given, and the rigid parts are exposed in the images of all the sub-division.

3. A movement of the object in its natural coordination is performed in video sequences, which are used by the algorithm to perform the following operations: (i) shift the previous image to the current image, (ii) move the object along the coordinate axes, (iii) register the relative movement of the rigid parts, and (iv) sort new images into clusters.

4. The algorithm aims to construct a model of the object based on the above steps. However, it is important to note that the algorithm only works with images captured by the camera in such a way that most of the surface is visible at any given time.

The algorithm is designed to work with images captured by the camera in such a way that most of the surface is visible at any given time. This allows the algorithm to construct a model of the object based on the above steps. However, it is important to note that the algorithm only works with images captured by the camera in such a way that most of the surface is visible at any given time.
The algorithm is presented in the form of a thesis:

**Technion - Computer Science Department - M.Sc. Thesis - MSC-2008-06 - 2008**

Figure 10 shows the algorithm's success in capturing the movement of the objects in real-time and accurately reconstructing the 3D models of the objects. The examples displayed in Figure 10 are based on the movement data obtained from standard cameras. The author, a student of the Computer Science Department, used MATLAB to implement the algorithm.

The algorithm is a powerful tool for analyzing and understanding the movement of objects, especially in the field of computer vision. It can be used in various applications, such as robotics, computer-aided design, and animation.

The results show that the algorithm can accurately track the movement of objects in real-time, even in complex environments. This is achieved through the use of advanced algorithms and hardware. The algorithm is flexible and can be adapted to various applications, making it a valuable tool for researchers and developers in the field of computer vision.

In conclusion, the algorithm presented in this thesis is a significant contribution to the field of computer vision. It provides a powerful tool for analyzing and understanding the movement of objects, and it has the potential to revolutionize the way we approach computer vision problems.

**References**

1. MATLAB
2. [MA06] ANN
המחкер עובר בניהית פروف' חיים גוטסמן בפקולטה למודי המחשב.
bilt מודל ושחזור תנועות חלקי של גופים

מברס מנדל מיריצי מזלמה עצום

מרתיע על מחקר

לשם מודל חלקי של תנועות לדגרי ממשלת התותח

מגיסר למודיעים במדעי המחשב

יורי פוקלי

ורשע לתקון מעבדות - מכון מיסקולר ליאוניאל

حذر תשמ"ח חсим

מרץ 2008
בנייהモデル ושחזור תנועת גופים שלגופים

מבוסס שלד מירדיאו מחלמאת טופמק

יורי פקלני