Fast and Accurate Closest Point Search on Triangulated Surfaces and its Application to Head Motion Estimation

DENNIS MAIER, JÜRGEN HESSER, REINHARD MÄNNER
Lehrstuhl für Informatik V
Universität Mannheim
B6, 23-29C, 68131 Mannheim
GERMANY

Abstract: - The Iterative Closest Point (ICP) algorithm is widely used to register two roughly aligned surfaces. Its most expensive step is the search for the closest point. Many efficient variants have been proposed to speed up this step, however they do not guarantee that the closest point on a triangulated surface is found. Instead they might result in an approximation of the closest point. In this paper we present a method for the closest point search that is fast and exact. The method was implemented and used to evaluate the accuracy of head motion estimation using dense 3D data obtained from a stereo camera system. The results show that the accuracy of the estimated head motion is better than 1 mm for translational movements and better than 1 degree for rotations.

Key-Words: - Iterative Closest Point, 3D Registration, Stereo, Head Motion Estimation

1 Introduction

We consider the problem of head motion estimation using dense 3D data obtained by a stereo matching algorithm using a calibrated camera system.

The Iterative Closest Point (ICP) algorithm proposed by Besl and McKay [1] is a standard solution to register two roughly aligned 3D point sets, \( P_1 \) and \( P_2 \). Its three basic steps to compute the Euclidean motion that aligns \( P_1 \) with \( P_2 \) are: 1. pair each point of \( P_1 \) to the closest point in \( P_2 \); 2. compute the Euclidean motion that minimizes the mean squared error (MSE) between the paired points; 3. apply the transformation to \( P_1 \) and update the MSE. The three steps are iterated until the MSE falls below a certain threshold or a maximum number of iterations is reached. It has been proven that the algorithm converges in terms of the MSE. Robust statistics like least median of squares and least trimmed squares have been used to make the algorithm robust against outliers [10][2]. The accuracy of the algorithm can be improved when using points on surfaces [8] instead of point clouds.

A recent overview of efficient variants of the ICP algorithm can be found in [7]. The most computationally expensive step in the ICP algorithm is finding the closest points. Simon [9] uses k-d trees and closest point caching to speed up this step. The k-d tree facilitates fast lookup of the closest vertex of the triangles of a surface to a given point. This is followed by a local search in the triangles that contain this vertex.

The Grid Closest Point algorithm (GCP) proposed in [6] uses binning to find the closest point. The space is devided into cubes and for each cube the distance of its center to the surface is stored. An approximation to the closest point of a point \( p \) is then found by determining in which cube the point \( p \) lies and adding to its distance from the center of the cube the stored value of the cubes distance to the surface.

Another method is to project the point onto the destination mesh, then follow this by a neighbor-to-neighbor walk in the destination mesh that attempts to find the closest point [7]. This method might fail to deliver the true closest point if the destination mesh contains holes or is not connected everywhere.

The existing methods do not guarantee to find the true closest point. They either implicitly yield approximations to the closest point or make assumptions, such as the mesh does not contain holes, that might not be fulfilled when dealing with 3D data resulting from stereo algorithms. Since we want to register two surfaces reconstructed with a stereo algorithm, we are interested in accurate closest points on triangulated surfaces.

The contribution of this paper is twofold. First, we develop a data structure to eliminate as many triangles as possible from the closest point search. A method is proposed that allows efficient search of the true closest point in an ICP algorithm using this data structure. The method is then used in an ICP algorithm to register two surfaces. Second, the accuracy of head motion estimation using a stereo camera system to acquire 3D data is evaluated with a special mechanical setup for measurements and a head phantom.

The paper is organized as follows. In section 2 we describe our method to search the closest point on a triangulated surface. The 3D data acquisition is explained in section 3 and the measurement setup is described in section 4. In section 5 results of the head motion computation are presented followed by conclusions in section 6.
2 Closest point search

A brute force method to find the closest point on a triangulated surface would be to project the given point onto every triangle of the surface and then choose the projected point with the minimum distance. To speed up the process we try to minimize the number of necessary projections. The idea is that one can classify many triangles of a surface to contain only points that have a greater distance from the given point than another point on the triangulated surface. We thus build a data structure that contains geometrical information of the surface that allows to classify as many triangles as possible irrelevant for the closest point search. The closest point must then lie in one of the remaining triangles and only those must be evaluated. Figure 1 shows a triangle mesh and a 3D point for which the closest point on the surface is searched. The point is projected on those triangles marked bold and the projection with minimum distance is the closest point. The other triangles were classified as having a greater distance to the point than at least one other triangle of the surface and can be eliminated from the search.

Figure 1: Closest point search on triangle mesh.

We now describe the data structure that enables us to classify the triangles into those that are irrelevant for the closest point search and those that must be examined. We divide the space into not necessarily disjoint subspaces that contain part of the surface. We use spheres as subspaces. For a given point we compute its distance to the center of a sphere. This distance plus the radius of the sphere is the upper bound for the distance of all points contained in the sphere to the given point. To get a lower bound for all points in a sphere we substract the radius of the sphere from the distance of the center and the given point. If the lower bound for sphere $S_1$ is greater than the upper bound of sphere $S_2$, all points contained in $S_1$ can be classified as irrelevant for the closest point search.

The spheres are organized in a tree to allow fast classification. For every triangle in the surface, we generate the smallest sphere that contains this triangle. The set of these spheres represent the first level of our sphere tree, see fig. 3 on the left. We then successively merge spheres to generate higher levels of the tree. We define a distance for two sphere nodes $S_1$ and $S_2$:

$$d_S = \frac{r_{12}}{r_1^2 + r_2^2}$$  \hspace{1cm} (1)

where $r_1$ is the radius of $S_1$, $r_2$ the radius of $S_2$ and $r_{12}$ the radius of the smallest sphere containing $S_1$ and $S_2$. If two sphere nodes have a distance $d_S$ smaller than a certain threshold the spheres are merged. Every level of the tree has its own threshold.

Figure 3 shows four levels of the sphere tree over a synthetic surface from fine to coarse.

To find the triangles that must be examined for containing the closest point to a point $p$, we compute for every sphere of the highest level of the tree

$$d(p, c_i, r_i) = \|p - c_i\| + r_i$$  \hspace{1cm} (2)

where $c_i$ is the center of the sphere $S_i$ and $r_i$ its radius. We define $d_{min}$ to be the minimum of all those values. Then we examine all spheres of the highest level again, and check if $\|p - c_i\| - r_i < d_{min}$ in which case the sphere cannot be classified irrelevant and so we check the criteria recursively for all its children, always updating $d_{min}$. When we reach a leaf node which cannot be eliminated it is added to a list $L$ for later examination. As the recursive process continues $d_{min}$ might be reduced further, making it possible to eliminate some of the elements of $L$, which were added when $d_{min}$ was still large. Therefore, at the end of the recursive process we eliminate all elements from the list $L$ that can be classified irrelevant with the final $d_{min}$. Then, the point $p$ is projected onto all elements in the list $L$ and the one with the minimum distance is the closest point.

Figure 2: Projection of a point using the sphere tree.
Figure 2 shows the projection of a point onto a surface using the sphere tree. On the top, the remaining spheres of two different levels of the sphere tree are shown. The final result of the tree search is shown in the bottom of fig. 2. All triangles in the shaded area must be examined for closest point.

The tree is built from bottom to top. A working list contains all sphere nodes of the current highest level. At the beginning, the working list consists of all minimal spheres that contain one triangle of the surface. When two spheres are merged, they are deleted from this working list, added as children to the merged sphere node and the merged node is added to the working list.

The merging process always works on the current top level of the tree. It is possible that one node contains more than one node of the next lower level. Because the spheres are successively merged pairwise as described before in a certain order it cannot be guaranteed that the resulting sphere is the smallest possible sphere containing all its children, see fig. 4. After no more spheres can be merged with sphere $S$, one could compute the smallest possible sphere $S'$ containing all its children. However, this is not done in our implementation due to the computational overhead and the usually small overestimation.

3 Obtaining dense 3D-data

Our imaging system consists of two analog Sony color CCD cameras EVI-370 with 12 times zoom, and autofocus lens, see figure 5. These cameras can be synchronized to guarantee that the images of the two cameras are taken at the same time instant. For the measurements of this paper the position of the right camera and the power supply were exchanged, the two cameras were then positioned at 6cm of each other.

The cameras produce images in PAL format with a resolution of 720 × 576 pixels. Since we want to measure movements we cannot make use of the interlaced full frame where the even and odd fields correspond to a different point in time. To make sure that all pixels in one image are taken at the same time, we therefore grab only half-frames. In order to keep the aspect ratio only every second pixel of a scanline is selected. Therefore, we end up with an image of size 360 × 288. The images are captured using our own FPGA frame grabber. The system is calibrated using a standard calibration. The lens distortion in the images is corrected and the stereo pairs are then rectified [11] to allow efficient calculation of disparity maps. We use the correlation based algorithm described in [3] to match the stereo pairs. Because faces usually have little texture content, we optionally improve the disparity maps with the anisotropic stereo algorithm of [4] when there are large unmatched regions. Note however, that due to self occlusion at the nose, the final disparity map might still contain holes. Figure 6 shows on the left a disparity map computed from a stereo pair of our head phantom and on the right the 3D reconstruction.
4 Measurement system

As specimen we used a styrofoam head phantom. This phantom was mounted on a plate to be fixed. In order to measure the translation accuracy, we mounted two tracks on another board whereby the tracks are such tight that the plate fits in without gap. The measurement was then performed using a caliper rule that measures the distance of the plate at an initial position, defined by a wooden bar, and its final position. This setup can be used to measure movements in $x$- and $z$- direction by just turning the plate with the phantom by 90° (see fig. 7).

To measure the rotation accuracy, we used two boards that are combined by a hinge. One board is laid on the table whereby on the other one we mounted the plate with the phantom. By accurately defining the position of the hinges we can measure the angle by defining the distance between both boards at the corner of the upper one.

The other rotation angle is measured by drilling a bore hole into the plate with the phantom and mounting this plate on an axis going through this hole.

5 Results

We performed a number of measurements to assess the quality of the head motion estimation. Each measurement was run several times, then average and standard deviation were computed. The results can be found in column 2 and 3 of table 1. The first column always lists the true head movement according to our measurement device. The results show, that the motion estimation using dense data is very accurate and the standard deviation of the results is low. In cases where the results differ most from the true values, the 3D reconstruction was more noisy than in cases with results closer to the true values.

Although the accuracy of the dense method is very high, we observe that it is not significantly better than the sparse feature based methods of a previous investigation presented in [5]. The results of the method yielding the best results in [5] can be found in column 3 and 4 of table 1. The dense method presented in this paper is computationally much more expensive than the sparse methods. However, in some face images it might not be possible to automatically extract enough feature points to allow accurate motion estimation. The motion estimation method based on ICP to register two dense surfaces does not have this problem, the correspondences are established by closest point search.

Figure 8 shows the reconstruction of the reference head and the reconstructed point cloud of the translated head with vectors pointing to its correspondences before (left) and after (right) registration. The algorithm converged in about 15 iterations in our measurements except for the translations along the $x$-axis where more than 50 iterations were necessary. For the pure trans-
Table 1: Measurement results.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>ICP avg</th>
<th>ICP stddv</th>
<th>Feature avg</th>
<th>Feature stddv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translation along x-axis: 3 mm</td>
<td>2.73 mm</td>
<td>0.33 mm</td>
<td>2.74 mm</td>
<td>0.03 mm</td>
</tr>
<tr>
<td>Translation along x-axis: 5 mm</td>
<td>4.43 mm</td>
<td>0.19 mm</td>
<td>4.76 mm</td>
<td>0.07 mm</td>
</tr>
<tr>
<td>Translation along x-axis: 7 mm</td>
<td>6.50 mm</td>
<td>0.18 mm</td>
<td>6.89 mm</td>
<td>0.03 mm</td>
</tr>
<tr>
<td>Translation along x-axis: 9 mm</td>
<td>8.81 mm</td>
<td>0.22 mm</td>
<td>8.94 mm</td>
<td>0.04 mm</td>
</tr>
<tr>
<td>Translation along x-axis: 11 mm</td>
<td>10.58 mm</td>
<td>0.19 mm</td>
<td>10.93 mm</td>
<td>0.04 mm</td>
</tr>
<tr>
<td>Translation along z-axis: 3 mm</td>
<td>3.02 mm</td>
<td>0.22 mm</td>
<td>3.12 mm</td>
<td>0.2 mm</td>
</tr>
<tr>
<td>Translation along z-axis: 5 mm</td>
<td>4.48 mm</td>
<td>0.09 mm</td>
<td>5.25 mm</td>
<td>0.34 mm</td>
</tr>
<tr>
<td>Translation along z-axis: 7 mm</td>
<td>6.66 mm</td>
<td>0.27 mm</td>
<td>7.03 mm</td>
<td>0.30 mm</td>
</tr>
<tr>
<td>Translation along z-axis: 9 mm</td>
<td>8.45 mm</td>
<td>0.25 mm</td>
<td>8.79 mm</td>
<td>0.23 mm</td>
</tr>
<tr>
<td>Translation along z-axis: 11 mm</td>
<td>10.48 mm</td>
<td>0.45 mm</td>
<td>11.0 mm</td>
<td>0.17 mm</td>
</tr>
<tr>
<td>Rotation around x-axis: 3.65°</td>
<td>3.59°</td>
<td>0.12°</td>
<td>3.28°</td>
<td>0.29°</td>
</tr>
<tr>
<td>Rotation around x-axis: 5.60°</td>
<td>5.62°</td>
<td>0.14°</td>
<td>5.59°</td>
<td>0.69°</td>
</tr>
<tr>
<td>Rotation around x-axis: 9.10°</td>
<td>8.66°</td>
<td>0.07°</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Rotation around y-axis: 2.68°</td>
<td>2.69°</td>
<td>0.29°</td>
<td>2.58°</td>
<td>0.08°</td>
</tr>
<tr>
<td>Rotation around y-axis: 5.77°</td>
<td>5.96°</td>
<td>0.60°</td>
<td>5.66°</td>
<td>0.27°</td>
</tr>
<tr>
<td>Rotation around y-axis: 10.11°</td>
<td>9.51°</td>
<td>0.08°</td>
<td>10.90°</td>
<td>0.40°</td>
</tr>
</tbody>
</table>

6 Conclusions

In this paper we presented a method to accelerate the search for the closest point on a triangulated surface. This is used to speed up the most expensive step of the ICP algorithm. The presented data structure allows to eliminate many triangles of the surface from the search for the closest point. If many surfaces must be registered with a reference surface, as is the case in our targeted applications, the data structure must only be build once, which can be done in advance.

We evaluated the accuracy of the head motion estimation using dense 3D data obtained by a stereo camera system. The accuracy is better than 1mm for translation and better than 1° for rotation.

References