Rigid ICP registration with Kinect

**Students:** Yoni Choukroun, Elie Semmel

**Advisor:** Yonathan Aflalo
Overview

The main goal of the first part of the project was to perform an Iterative Closest Point registration on two depth maps obtained using the Kinect depth sensor in C++ on the windows platform.

The other purposes of this first part was to learn how to integrate alone big libraries (dynamic or not) to the project and to handle with the difficulties of implementing an algorithm on the different classes of the libraries whom do not match necessarily one with the other.

The second part of the project was to bond, two by two with the precedent algorithm, different scan frames get by the Kinect with the help of its motor to get a whole body depth image.

Development of the project

After reading the papers of Hao-li, Besl & McKay and Rusinkiewicz & Levoy we first tried, with different interfaces, to obtain the two frames of the depth map. We chose the OpenNi interface to get the wanted depth frames.

To implement the rigid ICP algorithm we had to use a library matrix handler, after searching some libraries (armadillo and others), we used (by Yonathan guidance) the OpenCV library for the matrices calculation need and the depth images displaying.

Then we developed the simplest rigid Iterative Closest Point algorithm (as describe below) same as in the Besl and McKay’s paper.

Thereafter, in order to ensure the success of our project we decided to improve our algorithm as describe below.

Then we had to control the motor, which had the consequence to back to a previous version of OPENNI (from 2.0 to 1.5, the development of this tool may have been stopped when PrimeSens get bought by Apple) and learn how to move it for our purpose.

After the fusion of the scans we decided to save the results in some formats to allow the user choose the interface he wants to process the depth results (for MeshLab and Matlab process/display)
Papers

The first paper we read was the Hao-Li one. The purpose of his paper was to reconstruct with a Kinect a full body in 3-D with the multiple frames of the body got with the Kinect motor and the rotation of the body on itself to scan all the parts of the body.

The reconstruction was divided in pipeline's stations where the first one: scanning, fusion and segmentation were the goal of the entire project. To perform this first part we need, after the body scan, to use the rigid ICP algorithm to bond the different frames together to get the whole body depth image.

Thus we had to implement the rigid ICP.

The second was Besl & McKay's paper. In this paper all the theory of the first (rigid) iterative closest point algorithm is described. After some mathematical preliminaries the paper focused on the representation of the data as the one which interested us: the Point sets (as the Kinect) and it describes the Corresponding Registration to apply.

The algorithm is based on two main actions on a picture. Given a model shape and a measured shape, we will apply on each point of the measured data set a rotation and a translation in the space. The goal is to optimize those transformations by minimizing the distance square error of the transformation on the measured shape to the model as:

\[ f(q) = \frac{1}{N_p} \sum_{i=1}^{N_p} \| \tilde{x}_i - R(q) \tilde{p}_i - \tilde{q} R \|^2. \]

In order to not affect the performances we cannot pass on all the data points of the images, then we have to select a part of them randomly.

To the algorithm understand where the two shapes (points cloud) are in the space we first calculate the center of mass of the two given shapes and get the cross covariance matrix to know what is the link between the two shapes points in the space. After optimization calculations we are able to obtain a unit quaternion vector which will give us the rotation transformation. Then after the rotation transform we can apply the translation transform relatively to the center of mass of the model. We can calculate now the new mean square error and check if the performances are satisfying.

The third paper was the paper of Rusinkiewicz & Levoy. This paper checked the performances of the different variants of the rigid ICP algorithm. We then checked which variance will be good enough for our purpose.

ICP algorithm is divided in five main steps.

Selection of points: we deduced from the paper and the supposed shape of the body (like the "wave" scene) that the random sampling give a good convergence but we also implemented the uniform sampling method which were supposed to improve only a little the performances but in our case the improvement was really important.
Matching points: the closest point algorithm (as described in Besl & McKay paper) gives a reasonable convergence rate for shapes like bodies.

The weighting and rejecting of pairs and the error metric minimization were not used in our algorithm, the initial rigid ICP paper did not treat those variants.
**Project algorithm**

Using the OpenNI interface we can get two depth frames from the Kinect. Threads assure the user to see itself and let him time to stand in front the camera where to get a good scanning of the body.

To get the frames we had to control the kinect motor. After several research about the properties about the motor, we find that it can tilt the field of view using the tilt motor in the sensor. The motor allows an additional \([+/-31]\) degrees.

The frames are converted to OpenCV Mat type to deal with displaying, filtering and registration on the depth images.

After getting the two frames, we will filter the depth image by distance: all the objects far from this distance and then the useless background will be deleted (zero value pixel). This step is also in the Hao-Li paper (part 3.2) but in our case any sophisticated segmentation algorithms were used but only "distance" filtering.

Then the floor have not been removed that is why the position of the user and the field of view of the Kinect must not scan the floor in any frames. If not the algorithm will fail for sure, sampled points won't match between the body and the floor.

As said above we have implemented the simple Besl & McKay ICP algorithm we will describe it here:

We first randomly/uniformly sampling a fixed number of points whom values are different from zero (just the interesting shape) in the two pictures. There possibly are the same pixels chosen many times in the case of the random point selection. For uniform sampling a grid has been used.

With the OpenCV matrices calculation tool we will be able to compute all the vectors, matrices, eigenvectors needed in the algorithm.

We first compute the closest point in the two pictures by calculating the minimum Euclidean distance between all pair of points (in the shape and the model).

Then we can calculate all the matrices needed and the transformations to apply on each one of the sampled points. We check if the algorithm's convergence is satisfying enough and we decide to continue or stop it (the convergence to a local minimum have been proved in the MacKay paper).

We keep the rotation and translation transformation of each iteration in an accessible list for the user.

Texturization: body reconstruction by Poisson reconstruction or triangles meshing have been done on Matlab but because both the not incredibly good results and the Matlab and not C++ implementation have convinced us to not add it to the project. (The code and results are existing and can be given if asked). The RGB representation needs a triangulation to be display that is why it is not a part of the results.
Results
Other results can be found at the end of the report.

Input Frames:

<table>
<thead>
<tr>
<th>First frame</th>
<th>20th Frame</th>
<th>50th Frame</th>
</tr>
</thead>
</table>

Output:

Matlab Result Displaying
MeshLab Result
MeshLab Result

MeshLab Result

MeshLab Result
Methods

Problem:
The main problem is the initial position of 2 frames. If they are not close enough the algorithm may converge to a local minimum instead of the global. That’s why the biggest challenge is to find the right match. The sampling is then the most important step.
The third problem is the depth frame return by the Kinect is of a poor quality which means that in addition of the transformation the matching point is difficult because the 2 frames are different and find correspondences can be a more difficult task.

Solution:
We start by improving the method of the sampling, from random to uniform. The algorithm passes on the entire matrix on a grid, by sampling only depth values greater than zero (not background).
The Openni Camera driver provides default camera models out-of-the-box with reasonably accurate focal lengths (relating 3D points to 2D image coordinates). They do not model lens distortion, but fortunately the Kinect uses low-distortion lenses (|k1| \sim 0.1), so even the edges of the image are not displaced by more than a few pixels. Since our application requires maximum accuracy from the Kinect’s 3D data we should to perform a rigorous calibration.
But finally, since our application would be used by different users with different Kinect we decided to perform a general, average calibration with an average intrinsic matrix calibration. More precise calibration will improve the performances and the final result but ask from the user a tiresome calibration step.
In each execution the algorithm, as said before, has to find a sequence of rotation and translation; that’s why we wanted to “help” the algorithm by giving him a prior rotation that it needs to perform. For that we had to know an average angle covered by the Kinect between two frames.

\[ R_x(\theta) = \begin{pmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{pmatrix} \]

Below we joined the rotation matrix for a given angle:
The Rotation of the motor is really not accurate. We noticed that the motor move are not the same (not same angle) at each iteration. So we decided to not “help” the algorithm with the first matrix rotation (this try did not really improve the results).
At the beginning, we tried to perform ICP between a frame and the result of the all precedent frames bonded together. But after some execution, the result, due to the accumulated noise, is far from the reality and the algorithm don’t find right match between the new frame and the precedent bonded result. So we found better to perform ICP between pair of frames. It is an application of the transitivity law: if the transformation between frame \(i\) an d frame \(i-1\) is optimal and the transformation between \(i+1\) and \(i\) is optimal too, then the transformation sequence between \(i+1\) and \(i-1\) is optimal:
\[ Tr_n(F_n) = R_n(F_n) + T_n = Tr_n(Tr_{n-1}(\ldots Tr_1(F_1))) \]

with \( F \) the frame, \( Tr \) the transformation, \( R \) and \( T \) the corresponding Rotation translation.

The last methods that we used to reduce noise accumulation was to add only the new data part of the new frame at the bottom of the picture.

**Performances**

Our ICP algorithm and the ICP algorithm in general does not give good performances for big moves of the shapes. Only little rotations can converge in a satisfying way. Translation is not a problem, the calculation of the center of mass give us quickly the translation to apply even for big distances.

One of the problems of the closest point matching part is that for rotation of the body the distance can match points in the two pictures that are totally different but because of the rotation are at the same depth distance. The big number of samples and the small moves of the user seems solve this problem.

As said above, for big moves ICP algorithm can give bad results, sometimes it would appear it converges to a wrong local minimum (as described in the Besl & McKay paper).

Even for big moves the algorithm converges quickly enough: from 1-3 iterations for small moves or simple shapes (like lines created artificially in the code) to about 10 iterations for big rotations (without or with translations) where it seems not converging well at all.

One of the implementation problems is the loss of data when converting from infinitesimal values of the vectors to indexes in the matrices for displaying that is the reason of the zero value pixel lines on the shape registration. One solution is to use a filter (median) but in some cases it can affects the displayed image (depends on the type of pixels around).

The bad resolution of the Kinect affects the image’s acquisition: holes in the target shape.

At the optimization step, we faced a tradeoff: good fusion result vs time. We test all our methods explain before with different parameters.

The number of sampling has the most critical influence on the performance, it’s varies between 8000 or less for bad results to 12000 or more for good results.

Another parameter was the threshold and the number of iteration in each ICP algorithm, after several test we fixed the threshold at 0.001.

As explain above for big rotation it seems not converging well at all, that’s why we decided to perform the algorithm in each frames and not by step of two, despite the fact that the time would have been then divided.

Then we can say that the average time for the ICP running step is between 1.2~3mn depending on the number of sampled points. The scanning and reconstructing time is about less than one minute. Finding a correspondence radius can improve the closest point search step but all is a matter of wanted performances.

Another critical point is the presence of noise in the picture. The resolution of the Kinect is bad and it returns sometimes irrelevant pixels whom are critical and make the algorithm fail.
About the project

The reading of the papers was very interesting and introduced us to the world of computer & geometric vision and to the academic space.

The lack of good materials (good computers) at the beginning of the project and the ability of solve computer maintenance problem both in hardware and software were very difficult for us, but thanks to google we gone through it (almost). We lost a lot of time by understanding by our own the way the libraries work and their inabilitys to deal with simple problems like casting in OpenCV in matrices conversions etc. the software problems did not help us too.

We think support the students on all this maintenance will save lot of time and let the students more time to improve the project itself and not to throw them to Google’s help to deal with those new problems never seen before in regular courses at CS or EE. Today we certainly know others libraries and tools would have make the project easier and better.

We would like to thank all the peoples from the Laboratory whom help us to complete this project.
References

"3D SELF-PORTRAITS"
Hao Li, Etienne Vouga, Anton Gudym, Jonathan T. Barron, Linjie Luo, Gleb Gusev


"A Method for Registration of 3-D Shapes"

IEEE Trans. on Pattern Analysis and Machine Intelligence (Los Alamitos, CA, USA: IEEE Computer Society)

"Efficient Variants of the ICP Algorithm"
Szymon Rusinkiewicz, Marc Levoy

Stanford University
More results

Result2-1

Result2-2