Registration of two scanned range images using k-d tree accelerated ICP algorithm

By Xiaodong Yan Dec. 2003

Overview

- Review different variants of ICP
- Our choice of variants at each step of ICP
- K-D tree structure
- Our approach to register two Range Image
- Brief algorithms and some note regarding implementation
- Future work

ICP review

- ICP algorithm has become the dominant method for aligning three-dimensional models based purely on the geometry of the meshes
- The algorithm is widely used for registering the outputs of 3D scanners, which typically only scan an object from one direction at a time

ICP review (continued)

- It mostly deals with rigid-body transform, using iteration to refine the transform by minimizing an error metric
- We will assume that an initial alignment is available. Next will focus on the effects of choice in the six stages of the ICP algorithm and choose a combination of approach for our implementation

Six Stage of ICP algorithms

1. Selection of some set of points
2. Matching these points
3. Weighting the corresponding pairs
4. Rejecting pairs
5. Assigning an error metric
6. Minimizing error metric

Selecting sets of points

a. Always using all available points

b. Uniform subsampling of the available points

c. Random sampling (with a different sample of points at each iteration)

d. Selection of points with high intensity gradient, in variants that use per-sample color or intensity to aid in alignment

e. Each of the preceding schemes may select point on only one mesh, or select source points from both meshes.

Selecting set of points(continued)

f. Choosing points such that the distribution of normals among selected points is as large as possible

We will try both approach *a* and *f* in our implementation

According to the result by [Rusinkiewicz 02] the approach *f* can reduce alignment error after more iteration (after 15 times), while others will stay the same for alignment error; choosing points from both mesh may also improve results when the overlap is small or when meshes contain many holes

Matching Point

- Find the closest point in the other mesh, may accelerate using a k-d tree and/or closest-point caching (our choice)
- Find the intersection of the ray originating at the source point in the direction of the source point's normal with the other surface
- Project the source point onto he destination mesh, from the point of view of the destination mesh's range camera
- Project the source point onto the destination mesh, then perform a search in the destination range image
- Combination of the above, compatibility metrics based on color and angle

Weighting of Pairs

- Constant weight
- Assigning lower weights to pairs with greater point-to-point distances
- Weighting based on compatibility of normals
- Weighting based on the expected effect of scanner noise on the uncertainty in the error metric

Weighting of Pairs (continued)

- Results from [Rusinkiewicz 02] shows that the weighting function on convergence rate will be small and highly data-depended.
- We will not using any of them at the first step of implementation, but will try some of them late to see which one fit out type of data.

Rejecting Pairs

- Rejection of corresponding points more than a given (user define) distance apart
- Rejection of the worst *n*% of pairs based on some metric (usually point to point distance)
- Rejection of pairs whose point-to-point distance is large than some multiple of the standard deviation of distances

Rejecting Pairs (Continued)

- Rejection of pairs that are not consistent with neighboring pairs, assuming surfaces move rigidly
- Rejection of pairs containing points on mesh boundaries

We will use the last approach because the cost is low and has good convergence

Assigning Error Metric

- Sum of squared distances between corresponding points (our choice, has a closed-form solution of transformation to minimize error metric)
- Above point-to-point metric, taking into account both the distance between points and the difference in colors
- Sum of squared distances from each source point to the place containing the destination point

Minimizing Error Metric

- Repeatedly generating a set of corresponding points using the current transformation, and finding a new transformation that minimizes the error metric
- Above iterative minimization, combined with extrapolation in transform space to accelerate convergence
- Performing the iterative minimization starting with several perturbations in the initial conditions, then selecting the best result

Minimizing Error Metric(continued)

- Performing the iterative minimization using various randomly-selected subsets of points, then selecting the optimal result using a robust metric
- Stochastic search for the best transform, using simulated annealing

We will be using the first approach and may test others later on.

K-D tree analysis

A k-d tree structure will be used to accelerate the iteration step to find the closest point, which cost long time. The algorithm to construct a k-d tree and search the nearest point is stated in [A. Moore 91].

Based on results from [Talbert & Fisher 00], we will change the original Moore’s implementation of Bentley's algorithm to get better results

Implementation steps

- Input: two scanned range data image A, B of a 3D object, (an initial approximate transform already moved the coordinates of A to a close position)
- Output: a unit quaternion of rotation (the multiple from each iteration) and vector of translation

Some implementation details

1. Build a mesh based on the data we have on A and B.
2. Construct a k-d tree on vertex of image B.
3. For each point on A find the nearest point on B. (an improved version will choice points from A confine with normal distribution)
4. Eliminate pairs in which either points is on the boundary of the mesh
5. Find the rigid transformation that minimizes a least-squared distance between the pairs of point (see below)
6. Iterate until convergence

Notes on **Finding the rigid transformation**

Given a set of pairs of points from A and B, to compute the rotation and translation can get by follow algorithm :

Coordinates of points in A and B are noted as $PTA_{x/y/z,i}$ and $PTB_{x/y/z,i}$

$$S_{xx} = \sum_{i=1}^{n} PTA_{x,i} PTB_{x,i} \qquad S_{xy} = \sum_{i=1}^{n} PTA_{x,i} PTB_{y,i}$$

$$N = \begin{bmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{bmatrix}$$

Rigid transformation (continued)

- The unit quaternion will be the eigenvector corresponding to the most positive eigenvalue of the matrix *N*.
- A4. section of [Horn 87] paper provide a nice method to solve the Eigenvector.
- With the quaternion now that now have the new position of the transformation. Based on this we can find the new set of coordinate for points in mesh A. Then we will be calculating the ***Ss*** and ***Ns*** until convergence.

References

[1] Grey Turk and Marc Levoy, Zippered Polygon Meshes From Range Images Proc.SIGGRAPH'94 In *Computer Graphics Proceedings*, 1994, ACM SIGGRAPH, pp. 311-318

[2] B. K. P. Horn, Closed-form solution of absolute orientation using unit quaternions, *J. opts. Soc. Amer*. A vol. 4, no. 4, pp. 629-642, Apr. 1987

[3] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3):145--155, 1992

[4] P.J. Besl, N.D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. on Pattern Recognition and Machine Intelligence*, Vol. 14, No. 2, pp. 239-256, 1992

[5] Szymon Rusinkiewicz, Marc Levoy, Efficient Variants of the ICP Algorithm *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, 2001

[6] Andrew Moore, An introductory tutorial on kd-trees *Tech. report Technical Report No. 209, Computer Laboratory*, University of Cambridge, Robotics Institute, Carnegie Mellon University, 1991

[7] Douglas A. Talbert, Douglas H. Fisher: An empirical analysis of techniques for constructing and searching k-dimensional trees. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* 2000: 26-33