*Regular Article*

# 3D Reconstruction using Kinect Sensor and Parallel Processing on Graphics Processing Unit

**Thuong Le-Tien[1], Marie Luong[2], Thai Phu Ho[1], Viet Dai Tran[1]**

[1] Ho Chi Minh city University of Technology, Vietnam
[2] Université Paris 13, France

Correspondence: Thuong Le-Tien, thuongle@hcmut.edu.vn

Manuscript communication: received 20 August 2012, accepted 14 June 2013

*Abstract*– **One of depth cameras such as the Microsoft Kinect is much cheaper than conventional 3D scanning devices, thus it can be acquired for everyday users easily. However, the depth data captured by Kinect over a certain distance is of low quality. In this work, we implement a set of algorithms allowing users to capture 3D surfaces by using the handheld Kinect. As a classic alignment algorithm such as the Iterative Closest Point (ICP) does not show efficacy in aligning point clouds that have limited overlapped regions, another coarse alignment using the Sample Consensus Initial Alignment (SAC-IA) is incorporated in to the registration process in order to ameliorate 3D point clouds' fitness. Two robust reconstruction methods namely the Alpha Shapes and the Grid Projection are also implemented to reconstruct 3D surface from registered point clouds. The experimental results have shown the efficiency and applicability of our blueprint. The constructed system obtains acceptable results in a few minutes with a low price device, thus it may practically be an useful approach for avatar generations or online shoppings.**

*Keywords*– **Kinect camera, 3D-image reconstruction, sample consensus initial alignment, point clouds, alpha shapes, grid projection.**

## 1 Introduction

Three-dimensional reconstructions of small individual objects has been researched over the last few decades and applied into quality assurance, games, facial representation, design, etc. Existing 3D scanning technology is often based on specialized complex sensors, such as structured light camera. Despite high quality output, they are expensive and require expert knowledge for operations, which is unapt for general users. On the other hand, if handy and cheap 3D scanners were more amenable, 3D shape models could become widely used.

Depth cameras such as the Microsoft Kinect - a compact, low-price, and easy-to-use video camera, have recently attracted much attention. Compared with conventional 3D scanners, they are able to capture depth and image data at video rate and have little consideration of the light and texture condition.

In this work, we use a 3D object scanning approach that can be used to obtain several viewpoints of models visualized as point clouds (sets of point in 3D space) acquired from a Kinect such that those point clouds must partially overlap others next to them. Users acquire point clouds by freely moving the camera around an object. After that, the point clouds are registered into a single complete model. Then it is fed into surface reconstruction algorithms to create 3D mesh. The algorithms are parts of the Point Cloud Library (PCL), an open standalone project for point cloud processing. This whole process is straightforward and user-friendly. With the cost of $150, Kinect is much cheaper than other 3D scanners.

The biggest challenge when realizing this process into practice is the matching level between different clouds. Moreover, the obtained raw data are not entirely stable due to occlusion and Kinect limited sensing. Another problem is the processing time, in which most reconstruction algorithms are greedy, thus our group aims to reduce it by implementing General-purpose computing on Graphics Processing Unit to speed up the whole process, which is divided into parallel tasks to be executed on GPU. The main contributions of this

Figure 1. A point cloud captured from Kinect

work can be summarized as follows:

1) Implementation of a registration method comprising coarse alignment and fine alignment,
2) Implementation of optimal methods for surface reconstruction,

3) Parallel processing on GPU to improve timing (computational cost),
4) WebGL visualization of 3D point cloud over web-browsers without requiring third-party plug-in at runtime.

In most 3D processing programs, ICP is the solely implemented registration method. However, it is susceptible to local minima and generally leads to divergence when overlapped regions of two registered point cloud are limited. Thus, in this work, we use ICP as fine alignment after coarse alignment (RANSAC algorithm) for a fast execution of the algorithm.

## 2 Related Works

The project Kinect-Fusion [1] displays impressive real time 3D reconstruction done at high frame rate and consists of triangle mesh generated in realtime providing high detail as this mesh is constantly refined with new capture. This project shows us a low-cost handheld scanning, and geometry-aware augmented reality and physics-based interactions. The author detailed a novel GPU pipeline that achieves 3D tracking, reconstruction, segmentation, rendering, and interaction, all in real-time using only a commodity camera and graphics hardware. This work using the Iterative Closest Point (ICP) algorithm for aligning the sequential frames and then apply many reconstruction method to create a rendered view of 3D scene.

The Intel research [2] did a sophisticated reconstruction system using Kinect in 2010. It features advanced large-scale reconstruction techniques like loop-closure, a global error-minimization technique beyond the scope of this project.

Nicolas Burrus did his own take on such a system inspired by Intel in his RGB-Demo [3], released in 2011. Its GPU-optimized registration was also released.

## 3 Kinect Sensor and Point Cloud Acquisition

The Kinect consists of an IR camera, an RGB camera, and an IR projector that casts a fixed speckle pattern [4]. Conversion of the pattern seen by the IR camera to a depth map takes place on the device, and depth map can be retrieved via OpenNI library. This depth data is processed in PCL I/O module, creating 3D clouds and additional color components.

The acquired point clouds from Kinect confront several 3D perception challenges: massive amount of data which can be up to $640 \times 480$ points, occlusions from peripheral objects that make up large parts of the clouds, irregular density and noise.

## 4 Algorithms

The entire 3D object reconstruction embraces two major phases: the registration phase includes filtering, segmentation, features estimation, registration; the remaining filtering and surface processing make up the reconstruction phase (Figure 2).

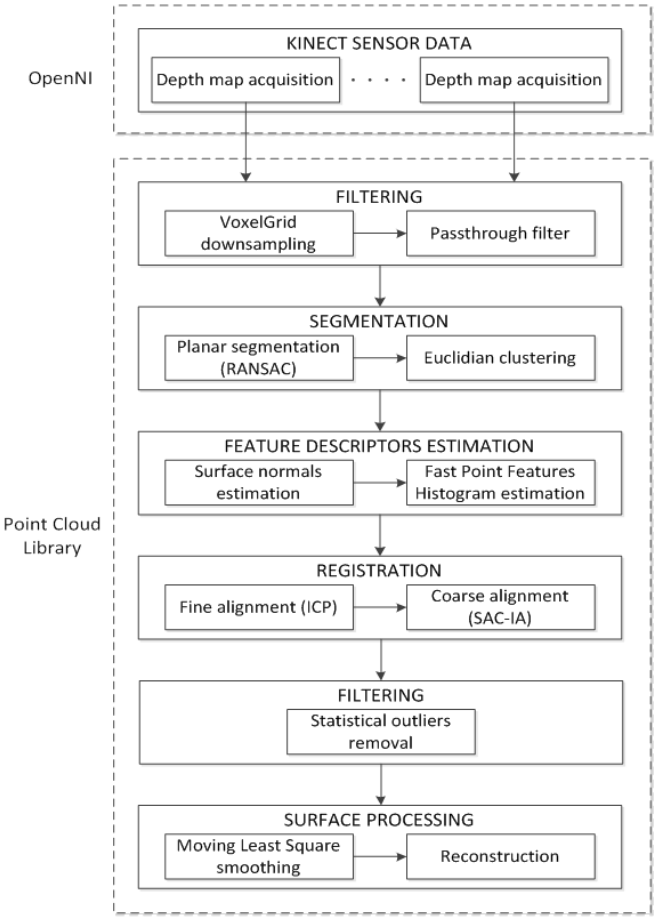


Figure 2. Workflow of the entire process

### 4.1 Filtering

*4.1.1 Downsampling:* As the number of points in a raw point cloud is relatively big ($640 \times 480$ points), the later processes will be slowed down. Since we are working with 3D data, voxelgrid filtering will be used. Voxel grid generation is used to divide 3D space into small 3D cubes. If the number of points inside a cube of specific dimension exceeds a certain threshold then the whole cube will be reduced to a single point at the points' centroid. There are many threshold will be shown in the following algorithms in the remain of this report. These parameters are chosen based on experimented method rather than a theoretical ones. For instance, we choose this certain threshold around 1 millimeter. With a smaller value of threshold, the rest points of input point clouds will be relatively big, and it does not bring the efficiency for later process.

*4.1.2 Passthrough filtering:* Acquired point clouds usually contain main object and surrounding area with peripheral objects such as floors, walls…Our goal is to eliminate superfluous objects using passthrough filter. Passthrough filter passes points in a cloud based on constraints for one particular field of the point type, for example, $z$ direction, aka. Kinect's viewpoint. It iterates

through the entire cloud once and filter non-finite points and points outside user-specified interval $[z_0, z_1]$.

Apart from these filters, a filtering method called statistical outlier removal is also used after the registration process.

### 4.2 Segmentation and Clustering

A point cloud is processed with planar segmentation to suppress the floor. Planar segmentation uses RANdom SAmple Consensus (RANSAC) algorithm [5] to estimate parameters of the planar model from the dataset.

Inputs of RANSAC are 3D point cloud $P$, tolerance threshold of distance $d_t$ between the chosen plane and other points, $\Omega_f$ - maximum probable number of points belonging to the same plane, minimum probability $\alpha$ of finding at least one good set of observations in $N$ trials.

The algorithm uses the following steps:

- Randomly select three non-collinear unique points $\{p_i, p_j, p_k\}$ from $P$,
- Compute the model coefficients from the three points $(ax + by + cz + d = 0)$,
- Compute the distances from all $p \in P$ to the plane model $ax + by + cz + d = 0$,
- Count the number of points $p^\star \in P$ whose distance $d$ to the plane model belongs to $[0, |d_t|]$.

Every set of points $p^\star$ is stored, and the above steps are repeated for $k$ iterations, where $k$ could be estimated using

$$k = \frac{\log(1-\alpha)}{\log(1-(1-\epsilon)^s)} \tag{1}$$

in which, $\epsilon$ is the probability of picking a sample that produces a bad estimate (i.e. outlier), $s$ is the number of chosen points to estimate the model parameters, evaluating the ratio between $\Omega_f$ and the total points $\Omega$ in the cloud.

In this section, the tolerance threshold is chosen about a few centimeters. A smaller threshold will make an increase in time processing, and a lager one, otherwise, will reduce the precision of the algorithm.

*Clustering*

In Figure 3, the floor has been subtracted from the original cloud $P$. However, some residual fragments still remain in the result $P^\star$, as they do not belong to the plane, or the distance threshold $d_t$ was set too tightly. To eliminate these fragments, the clustering process follows the segmentation process.

A cluster $C_i = \{p_i | p_i \in P\}$ is distinctive from the cluster $C_j = \{p_j | p_j \in P\}$ if $\min |p_i - p_j| = d_{th}$ (the value of $d_{th}$ is about a few centimeters). The algorithmic steps [6] separating a point cloud into different clusters are as follows,

- Create a kd-tree representation for the input cloud $P$
- Set up an empty list of clusters $C$, and a queue of the points that need to be checked $Q$,
- Then for every $p_i \in P$, perform the following steps:
  + Add $p_i$ to the current queue $Q$,
  + For every point $p_i \in Q$ do:
    - Search for the set $P_i^k$ of neighboring points of $p_i$ in a sphere with radius $r < d_{th}$,
    - For every neighbor $p_i^k \in P_i^k$, check if the point has already been processed, and if not add it to $Q$,
  + When the list of all points in $Q$ has been processed, add $Q$ to the list of clusters $C$, and reset $Q$ to an empty list,
- The algorithm terminates when all points $p_i \in P$ have been processed and are now part of the list of point clusters $C$.

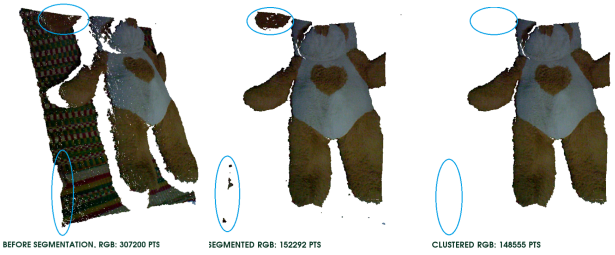


Figure 3. A point cloud before segmentation, after segmentation and after clustering

### 4.3 Feature Descriptors Estimation

Point clouds alignment requires correspondences between two partially overlapped pairs of point clouds. The features descriptor of the point clouds need to be extracted. Surface normals and curvature are such correspondences.

*Normals estimation*

The simplest method is based on the first order 3D plane fitting. The problem of determining the normal to a point on a surface is approximated by estimating the normal of a plane tangent to that surface, which becomes a least-square plane fitting estimation in [7]. The plane is represented as a point $x$ and a normal vector $\vec{n}$, and the distance from a point $p_i \in P_i^k$ to the plane is $d_i = (p_i - x)\vec{n}$. The values of $x$ and $\vec{n}$ are computed in a least-square sense so that $d_i = 0$. By taking:

$$x = \overline{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{2}$$

as the centroid of $P^k$, the solution for $\vec{n} = (n_x, n_y, n_z)$ is given by analyzing eigenvalues and eigenvectors of the covariance matrix $C$ of $P^k$:

$$C = \frac{1}{k}\sum_{i=1}^{k} \xi_i (p_i - \overline{p})(p_i - \overline{p})^T,\ C \cdot \overline{v_j} = \lambda_j \cdot \overline{v_j},\ j \in \{0,1,2\} \tag{3}$$

The term $\xi_i$ represents a possible weight for $p_i$, and usually equals 1. $C$ is symmetric and positive semi-definite, and its eigenvalues are real numbers $\lambda_j \in R$. The eigenvectors $\vec{v_j}$ form an orthogonal frame, corresponding to the principal components of $P^k$.

For the sign of $\vec{n}$, Kinect viewpoint $v_p$ is utilized. To orient all normals $\vec{n}$ consistently towards $v_p$, we need $\vec{n_t}(v_p - v_i) > 0$.

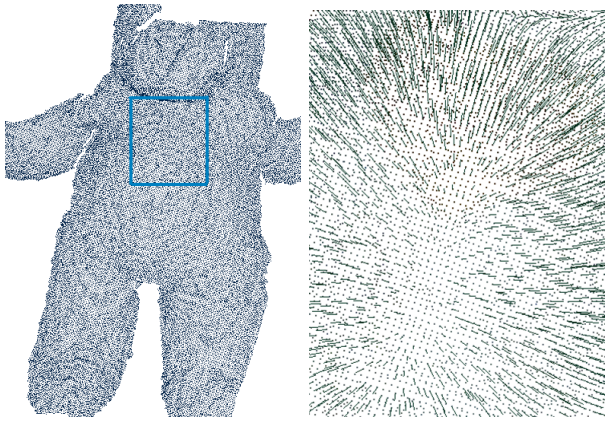

Figure 4. Normal estimation of a point cloud

The output surface curvature is estimated as a relationship between the eigenvalues of the covariances matrix as:

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{4}$$

*Fast Point Feature Histogram (FPFH)*

Surface normal and curvature cannot capture too much detail as they approximate the geometry of with only a few neighbors, resulting in similarities in feature values, reducing informative characteristics. To better feature descriptors, PFH - a powerful features extraction method, is used to group points on the same surface to a separate class.

For a source point $p_s$ and a target point $p_t$, the Darboux frame is $uvw$ (Figure 5):

$$\begin{cases} u &= n_s \\ v &= u\frac{p_t - p_s}{|p_t - p_s|} \\ w &= uv \end{cases} \tag{5}$$

Radu Rusu et al [8] suggested using $uvw$ frame to

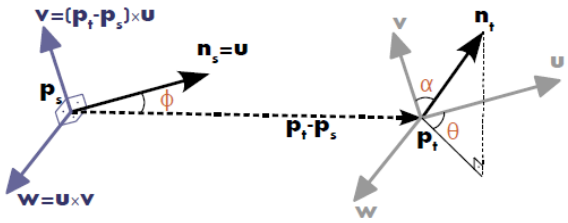


Figure 5. Representation of the Darboux frame and the angular PFH features for a pair of points $p_s$ and $p_t$ with their associated normals $n_s$ and $n_t$

calculate a set of angular features between the two normals $n_s$ and $n_t$:

$$\begin{cases} \alpha &= vn_t \\ \phi &= u\frac{p_t - p_s}{d} \\ \theta &= \arctan(wn_t, un_t) \end{cases} \tag{6}$$

where $d = |p_s - p_t|$ is the Euclidean distance between the two points $p_s$ and $p_t$. The PFH quadruplet $\langle \alpha, \phi, \theta, d \rangle$ is computed for each pair of two points in the $P^k$ neighborhood.

To create the final PFH representation for the query point $p_i$, all quadruplets is binned into a histogram. The binning process divides each feature's value range into $b$ subdivisions, and counts the number of occurrences in each subinterval.

In figure 5, $p_t$ is one of the k-nearest neighbors of they query point $p_s$. The number of quadruplets formed in a neighborhood $P^k$ is $C_k^2 = \frac{1}{2}k(k-1)$, with computational complexity of $O(k^2)$. For a point cloud dataset $P$ with $n$ points, the complexity is $O(nk^2)$.

To accelerate PFH, it simplified version FPFH [8] is suggested to reduce computational complexity to $O(nk)$:

- For each $p_q$, Simplified PFH (SPFH) triple $\langle \alpha, \phi, \theta \rangle$ between $p_q$ and its neighbors are computed as in (5)
- For each point its $k$ neighbors are re-determined, and neighboring SPFH values are used to weight FPFH of $p_q$ as:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k} \cdot FPFH(p_k) \tag{7}$$

where the weight $\omega_k$ represents a distance between the query point $p_q$ and a neighbor point $p_q$.

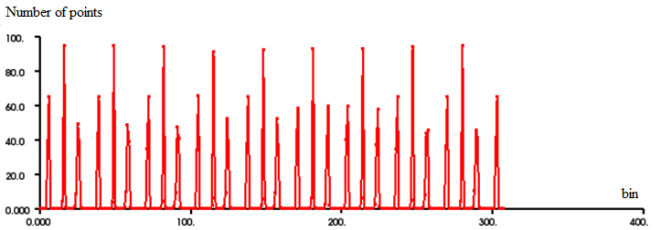


Figure 6. Fast Point Feature Histogram of a point cloud

### 4.4 Registration

Registration is to align point clouds into one complete object using estimated FPFH descriptors or correspondences. ICP is the solely implemented registration method but it has some disadvantages when overlapped regions of two registered point cloud are limited. Thus, ICP will be used as fine alignment after coarse alignment gets executed.

*Coarse alignment*

The algorithm searches for a set of corresponding points in the target point cloud for the persistent FPFH features of the source. Coarse alignment using SAC-IA [6] – [9] is based on a sample consensus formulation that samples large numbers of correspondence candidates and rank each of them quickly:

- Sample $n$ points $d_i$ from source: $|d_i - d_j| \geq \min d$
- For each $d_i$:
  - + Find $k$ closest matches in features
  - + Choose one to be correspondence $m_i$
- Estimation transformation $T = (R, t)$
- Filter inlier pairs with $(d_i^T - m_i)^2 < \epsilon$
- Loop $N$ times and pick $T$ having the most inliers.

*Fine alignment*

If $Q$ denotes a point cloud which is to be registered and moved to a static point cloud $P$, ICP [10] goes as following:

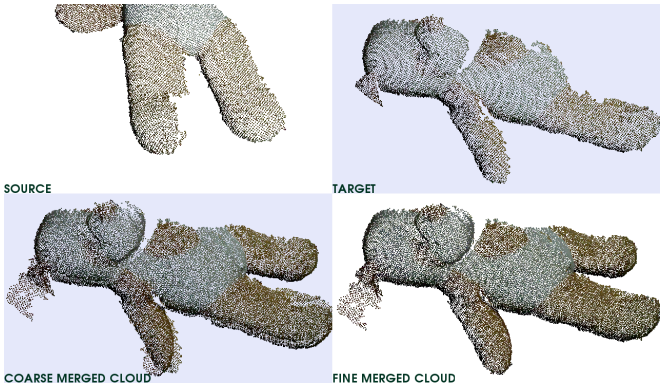


Figure 7. Comparison between point clouds after coarse and fine alignment. The RHS merged point clouds outweigh the LHS one in registration quality

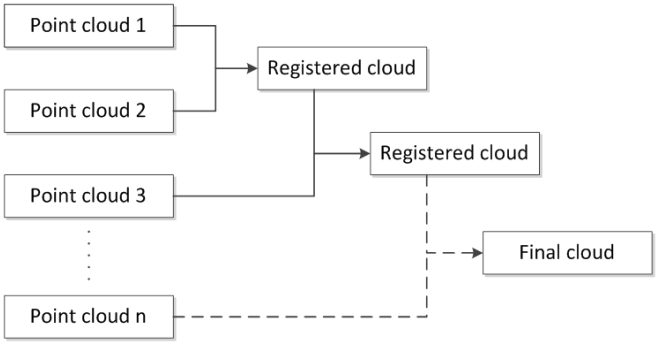


Figure 8. Illustration of pairwise registration

– For all points $q \in Q$, find the nearest neighbor $p \in P$.
– Find an orthogonal rotation $R$ and a translation $t$ minimizing the squared distances between neighboring pairs:

$$\min_{R,t} \sum_i \|(Rq_i + t) - p_i\|^2 \tag{8}$$

– Apply the transformation to $Q$.

This process is repeated over multiple runs until one termination criterion is reached. A means of ICP judgment is to use the Euclidian fitness score from two sets of correspondence distances:

$$\text{Fitness score} = \sum_i (Rd_i + t - m_i)^2 \tag{9}$$

To recap previous sections, figure 8 illustrates pairwise registration to form the complete model from various clouds.

### 4.5 Moving Least Squares Smoothing and Upsampling

In the merged cloud after pairwise registration (Figure 9), the density of points in overlapped areas is nearly two times higher than normal. Besides, there possibly are doubled regions. This leads to the use of Moving Least Squares [11] for smoothing and upsampling (Figure 10).

The first step is to compute a set of points $Q$ which are to be fitted to the surface. If only surface reconstruction is required, initial guess for $Q$ can be the final

Figure 9. Four input point clouds corresponding to four viewpoints and the pairwise-registered point cloud (bottom)

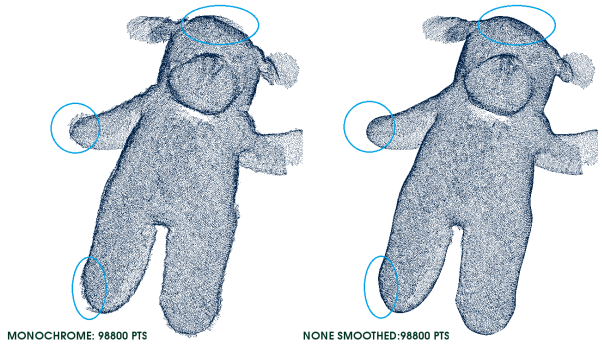


Figure 10. The cloud surface before and after MLS smoothing

cloud. For resampling, a set of equidistant grid points is generated in the proximity of the input cloud (the registered cloud) such that holes are covered. Holes-filling strategy [12] fills every hole that is smaller than a given radius. In the second fitting step, for each point $q \in Q, h = \mu_q + k\sigma_d, i = \overline{1,k}$, the coefficients are computed for its k-nearest neighbors in $P$:

$$w_i = \exp\left(-\frac{\|q - p_i\|^2}{h}\right) \tag{10}$$

The approximate surface is estimated via bivariate polynomial $f_{u,v} = \sigma_{i,j} u^i v^j$ from the original cloud surface $n_{u,v}$:

$$n_{u,v} = \sum_{j=1}^{x} c_j f_{u,v}^j = \mathbf{f}_{u,v}^T \mathbf{c} \tag{11}$$

The coefficients $c_j$ are computed via minimizing error function:

$$E(s) = \sum_{i=1}^{k} w_i \left(\mathbf{f}_{u,v}^T \mathbf{c} - n_{u,v}\right)^2 \tag{12}$$

Finally, $p_i \in P_f$ are projected to the approximated surface.

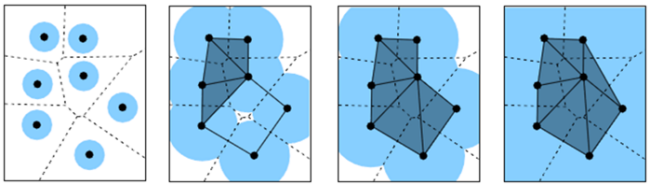

Figure 11. The alpha shapes of a seven-point set for different values of alpha. From left to right, top to bottom, alpha increases from zero to infinity

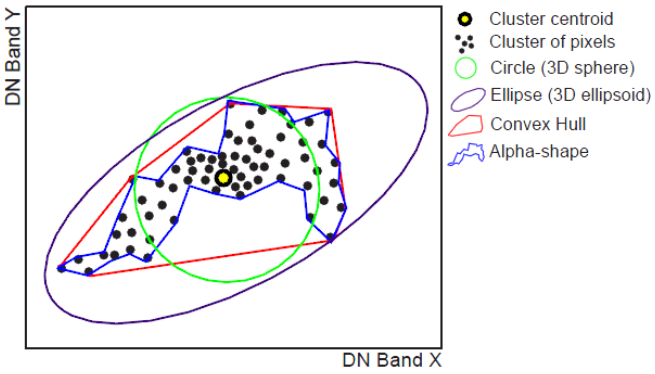


Figure 12. The alpha shapes of a point set

### 4.6 Surface reconstruction

Two main categories of surface reconstruction are mesh construction and implicit function.

*Alpha shapes* [13] belongs to mesh construction methods that preserve cloud surface and construct a polygon mesh linking the vertices. Alpha-neighborhood - a union of balls of radius $\alpha$ centered at each data point is formed (the circled region). The alpha-neighborhood is partitioned into cells by taking the intersection with the Voronoi diagram (the dashed lines). The alpha shape is the geometric dual of the partitioned alpha neighborhood (Figure 11). Alpha shapes simplifies to the point set when alpha tends to zero, converges to the convex hull as alpha tends to infinity (Figure 12). The reconstructed surface of the merged point cloud in figure 9 is display in figure 13.

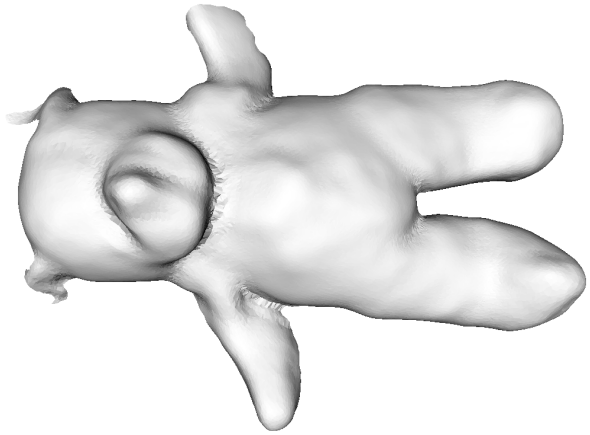

Figure 13. The surface formed by alpha shapes method

*Grid Projection* [14] – [15] belongs to implicit function methods that generate new vertices or polygon meshes

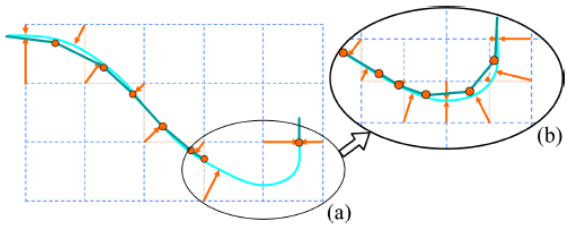


Figure 14. Grid projection approximates the extremal surface (blue curve) and constructs a simplicial surface (green curve) by examining orientation and magnitude of vectors (orange arrows) at two end points of each grid edge

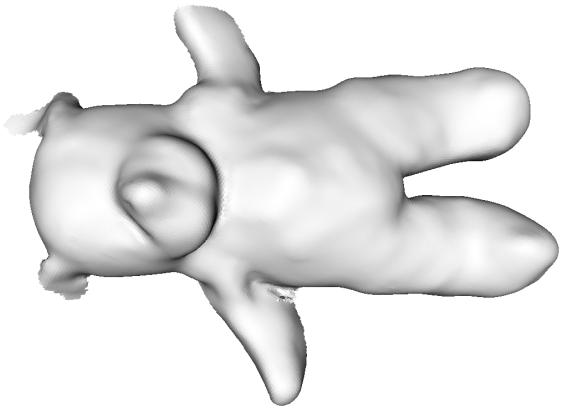

Figure 15. The surface formed by grid projection method

to as it fit the cloud surface into a smoother surface algorithm. Points are first partitioned into voxels, and a vector field is constructed, where the vectors at any given point are directed at the nearest point. A surface is determined by examining where vectors with opposite directions point towards. Edges in the voxels that this surface is reconstructed from are determined, and padding cells are also added. The center points of each voxel are then projected based on the edge intersections, and the surface is reconstructed by connecting these center points.

Many methods applied to 3D models imply a change of topology and geometrical distortion, which justifies the choice of Hausdorff metric to perform measurements over 3D space.

The distance between a point $p \in S$ and a surface $S'$ is:

$$d(p, S') = \min_{p' \in S'} \|p - p'\|_2 \tag{13}$$

with $\|\cdot\|_2$ is Euclidian norm. Hausdorff distance [16] is the distance from each point in surface $S$ to its k-nearest points in surface $S'$:

$$d(S, S') = \max_{p \in S} d(p, S') \tag{14}$$

## 5 Results

To judge registration quality, fitness score is used. A fitness score is considered good if it is below $10^{-4}$. It also depends on the percentage of overlapped area over total area of a cloud. In table , the fitness score drops 1.3 to 30 times when two-stage registration is used.

Table I
REGISTRATION FITNESS SCORE

| Model | ICP | SAC-IA & ICP |
|---|---|---|
| Bear (small overlap) | 0.0209211 | 0.00970484 |
| Bear (large ovl.) | 0.0448061 | 0.00150492 |
| Body (small ovl.) | 0.0304748 | 0.00125251 |
| Body (large ovl.) | 0.00126414 | 0.000950584 |
| Robot | 0.00612491 | 0.000639445 |
| Dinosaur | 0.00637339 | 0.000637991 |
| Shark | 0.00148293 | 0.000386612 |
| Minnie mouse | 0.00173132 | 2.24E-04 |
| Triceratops | 0.00457102 | 4.43E-05 |

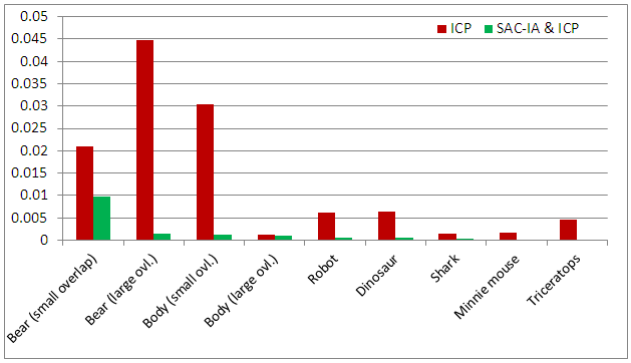


Figure 16. Fitness score comparison between ICP and SAC-IA & ICP

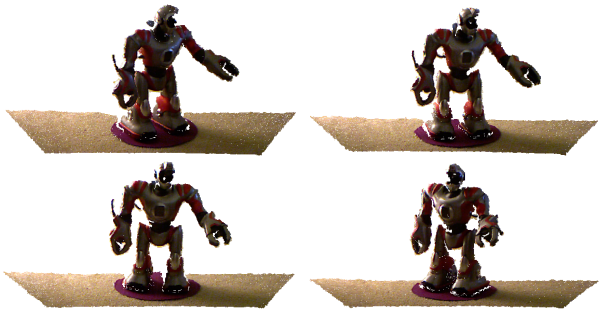

Figure 17. Four input point clouds of a robot model captured from Kinect

The fitness score get more improved when overlapped region of two consecutive point clouds gets large.

To assess surface quality, reconstructed surface should be compared to real surface, which is inapplicable. However, two1 largely overlapped point clouds can be finely registered into a template surface using ICP integrated in popular 3D processing programs. The Hausdorff distance is used to judge the reconstructed surface to the template.

In figure 19, mean Hausdorff distance between alpha shapes surface and real cloud surface is 1.46mm, while it is 1.87mm in case grid projection. Surfaces generated by alpha shapes are better than ones generated by grid projection. In term of timing, alpha shapes takes 15 seconds whilst grid projection requires up to 270 seconds for a typical 50,000-point cloud. Processing time of all processing tasks is visualized in figure 20 in case four input clouds are used. Registration cost is highest in time among all tasks. GPU implementation

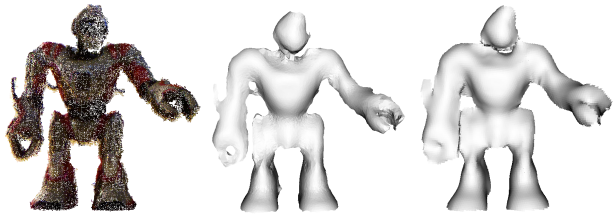

Figure 18. The pairwise registered cloud (left) of the four input clouds in figure 17, alpha shapes surface (middle) and grid projection surface


Figure 19. Hausdorff distance between alpha shapes surface (left, mean 1.46mm) and grid projection surface to real surface (right, mean distance 1.87mm)

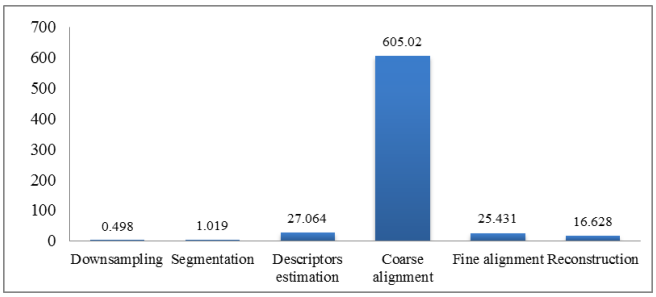


Figure 20. The pairwise registered cloud (left) of the four input clouds in figure 17, alpha shapes surface (middle) and grid projection surface

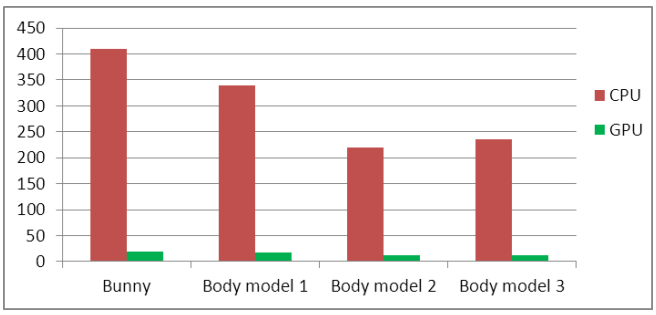


Figure 21. Registration time cost in second in CPU and GPU

of algorithms using nVidia GPU having modern CUDA architecture will exploit parallel threads to reduce time cost for the whole process.

Figure 21 exhibits the effectiveness in processing time cost for the heaviest task - registration, in which tested models consist of 20,000 - 50,000 points. In addition to a well-known bunny model, we acquired point clouds, using Kinect, from real objects such as three different body samples. The time cost is significantly decreased when we implement these tasks on GPU.

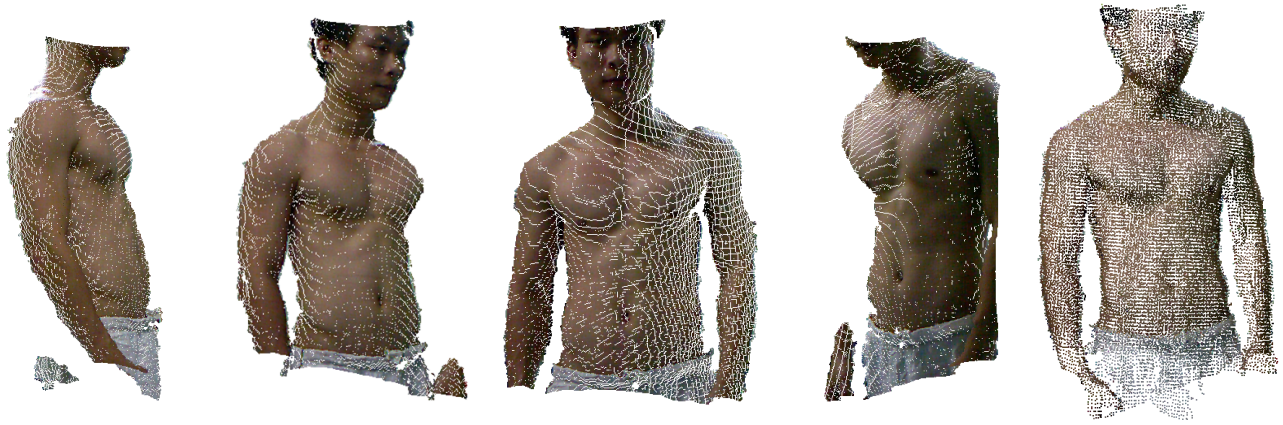

Figure 22. Four input clouds of a body sample and the pairwise registered model (right)

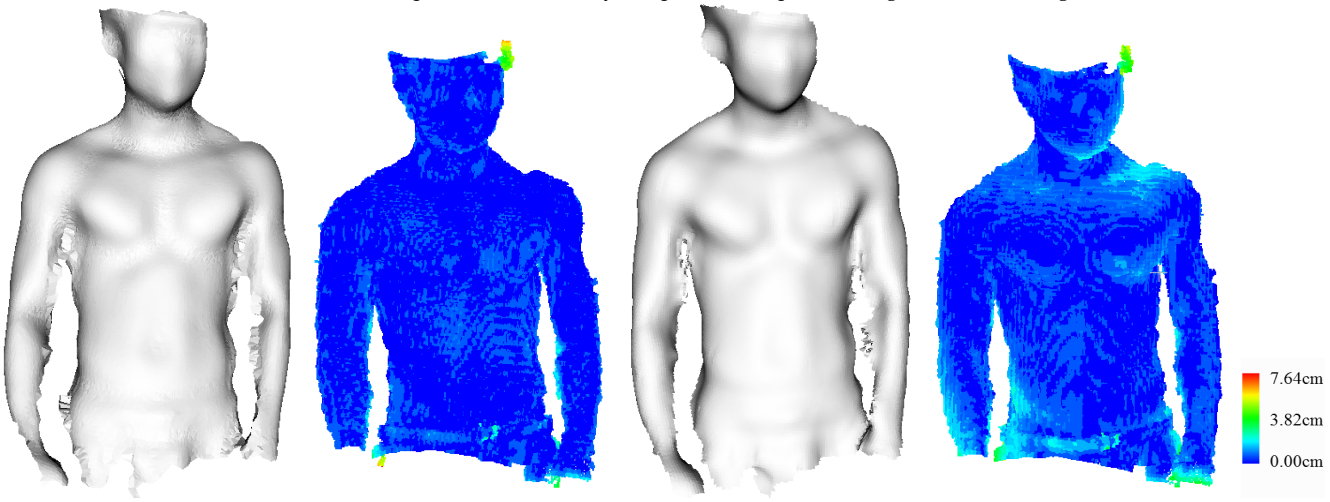


Figure 23. Alpha shapes surfaces and error plot (two lefts, mean distance 1.46mm) vs. grid projection and error plot (two rights, mean distance 1.73mm)

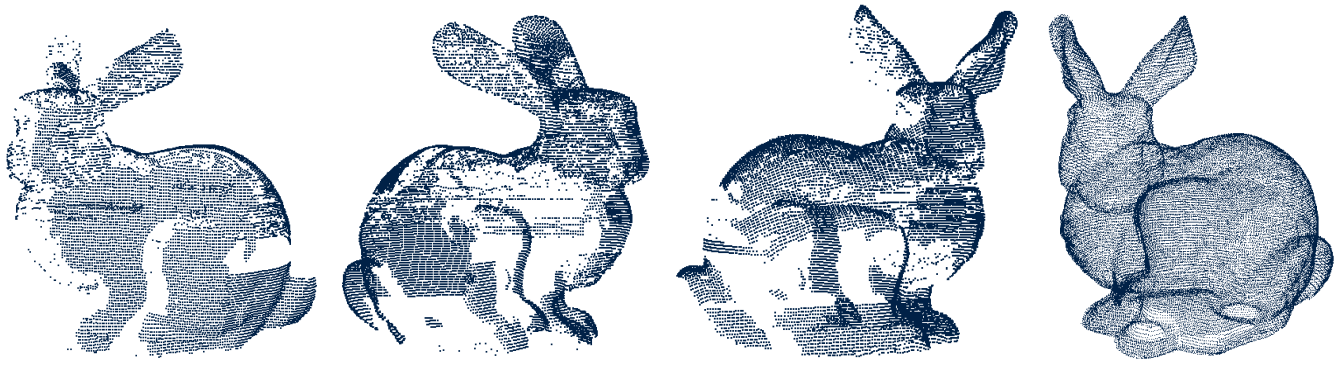

Figure 24. Three among eleven input clouds and pairwise registered cloud (right) of the bunny model

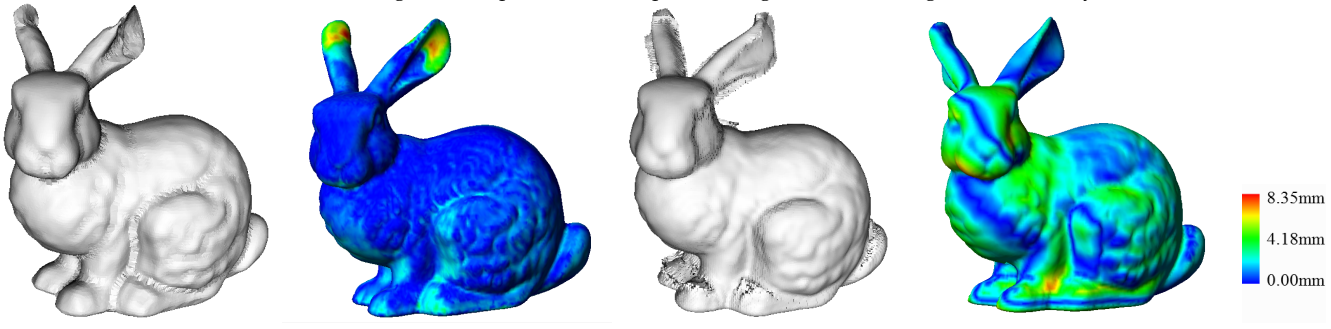


Figure 25. Alpha shapes surfaces and error plot (two lefts, mean distance 1.46mm) vs. grid projection and error plot (two rights, mean distance 1.73mm)

## 6 Conclusion

3D object scanning has so far had a limited range of applications due to expense, complexity, and space requirements. Here an acceptable accuracy reconstruction with an inexpensive commodity sensor is shown viable. We have demonstrated the feasibility of an object scanner that could work freely by combining point cloud datasets into a finely registered model. The key idea is to use dual alignment including SAC-IA and ICP algorithm to improve fitness score. Beside formidable alignment quality, two surface reconstruction algorithms were also implemented and proved to exhibit high quality mesh.

Future work should address the details of the surface of complicated objects. We believe that the implemented algorithm should work better with point clouds from sophisticated cameras. Optimization speed can be improved to make the work more interactive.

## References


[1] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2011*. IEEE, 2011, pp. 127–136.

[2] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[3] N. Burrus, "Kinect rgb demo," *Manctl Labs*.

[4] M. Viager, "Analysis of kinect for mobile robots," *Project Report for Electrical Engineering MSc Programme DTU*, 2011.

[5] M. Y. Yang and W. Förstner, "Plane detection in point cloud data," in *Proceedings of the 2nd int conf on machine control guidance, Bonn*, vol. 1, 2010, pp. 95–104.

[6] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, München, Techn. Univ., Diss., 2009, 2009.

[7] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008 (IROS 2008)*. IEEE, 2008, pp. 3384–3391.

[8] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE International Conference on Robotics and Automation, 2009 (ICRA'09)*. IEEE, 2009, pp. 3212–3217.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[11] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 544–552.

[12] J. Wang and M. M. Oliveira, "A hole-filling strategy for reconstruction of smooth surfaces in range images," in *XVI Brazilian Symposium on Computer Graphics and Image Processing, 2003 (SIBGRAPI 2003)*. IEEE, 2003, pp. 11–18.

[13] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 1, pp. 43–72, 1994.

[14] R. Li, "Surface reconstruction from point cloud," *Project Report at Washington University in St. Louis*, 2011.

[15] L. Phan, L. Liu, S. Abeysinghe, T. Ju, and C. M. Grimm, "Surface reconstruction from point set using projection operator," in *ACM SIGGRAPH 2008 posters*. ACM, 2008, p. 109.

[16] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance," in *Proceedings, IEEE International Conference on Multimedia and Expo, 2002 (ICME'02).*, vol. 1. IEEE, 2002, pp. 705–708.


**Thuong Le-Tien** was born in 1957 in Saigon, Vietnam. He received his B.Eng and M.Sc from the Ho Chi Minh city University of Technology (HCMUT), Vietnam then Ph.D degree from the University of Tasmania, Australia. He has been a lecturer at HCMUT since 1981 and is appointed to various academic positions at the University. He has been awarded the national associate professor title since 2003 then the national distinguished lecturer since 2008 in Vietnam and the distinguished invited professor from Mannheim University of Applied Science Germany, in 2009. His academic interests include Signal Processing, Electronics and Digital Communications.

**Marie Luong** received the Engineer Diploma from the Ecole Nationale Supérieure d'Electricité et de Mécanique de Nancy in 1991 and the Ph.D degree in Automatic, Diagnosis, Signal Processing from L'Institut National Polytechnique de Lorraine (Lorraine University), France, in 1996. She is an Associate Professor at Université Paris 13, Sorbonne Paris Cité, France, since 1998. Her research interests include image analysis such as restoration, super-resolution, segmentation and watermarking.

**Thai Phu Ho** was born in 1989 in Quang Nam province, Vietnam. He received his B.Eng from the Ho Chi Minh city University of Technology, HCMUT, Vietnam in 2012. His research interests are image processing, computer vision, wireless communication and networking.

**Viet Dai Tran** was born in 1989 in Binh Dinh province, Vietnam. He received his B.Eng from Ho Chi Minh city University of Technology, HCMUT, Vietnam in 2012. His research interests are image and video processing, digital communications and information theory.