Parallel Implementation of 3D Modelling of Indoor Environment Using Microsoft Kinect Sensor

Pawar Manojkumar
Department of Information Technology, NITK
NITK Surathkal,P.O. Shriniwasnagar
Manglore-575025,India
pawarmanoj89@gmail.com

G Ram Mohana Reddy
Department of Information Technology, NITK
NITK Surathkal,P.O. Shriniwasnagar
Manglore-575025,India
profgrmreddy@gmail.com

***Abstract*—3D visual modelling of indoor space will not only provide the detailed knowledge about the environment but also rich contextual information of the existing objects. In this paper, we propose a parallel implementation of 3D modelling of indoor environment using Microsoft Kinect depth camera. 3D maps are generated by Simultaneous Localization And Mapping [SLAM] technique. These 3-D maps will be more useful in Context aware and in Robotics applications.Iterative Closest Point [ICP] with initial guess by RANSAC is used for pair alignment.**

Many tasks of pair registration are parallelised using OpenMP and the performance evaluation of both sequential and parallel implementations are compared. Simulation results demonstrate that OpenMP based parallel implementation has achieved a speedup factor of 3.7 .

***Keywords*-3-D Modelling, SLAM, ICP;**

I. Introduction

3-D modelling is nothing but representing 3-D object in 3-D space with number of points or geometries. As in real life all objects are in 3 dimensions, so representing them in 3-D space provide much more information and realistic view about object. Earlier work mostly refer to modelling single object in 3-D space.

A 3-D model of an indoor environment include modelling of indoor space, floor, wall, objects inside room. These 3-D models will play more important role in context aware and robotics applications such as automatic route tracking, object detection. There are many 3-D modelling software available but most of them are designing software and it require designing skill to create model. Here we are going to propose a system which will automatically generate 3-D model from scan of indoor environment. Systems which are generating 3-D model from scanned data, most of them are using laser camera and robot to carry it. They are costly and maps generated from them are not realistic. A technical person is required to generate such models.

Kinect [1] is a motion sensing input device by Microsoft for the Xbox 360 video game console. It enables users to control and interact with the Xbox 360 without the need to touch a game controller, through a natural user interface using gestures and spoken commands. Kinect has RGB camera, Depth camera (infrared projector and receiver) and multi array micro-phone. Kinect is providing RGB and Depth information per pixel. Due to its functionalities, Kinect is used with PC for many applications. After the launching of depth cameras like Microsoft's Kinect or Prime Sense, it is possible to access RGB plus Depth data for each frame captured by single camera.

Here we are going to design a system using which if user scan his indoor environment with any depth camera, then the 3-D model of that environment will be automatically generated. A system is capable of generate map of rooms of 20 meter, with correct depth and colour information [2].

3-D localization and mapping is main issue for this type of system [3]. Localization is nothing but detecting and keeping track of camera trajectory. Many techniques can be used for this problem like GPS, Wi-Fi but due to their limitations and accuracy in indoor environment we are using technique which is extracting and comparing features of neighbouring frames to decide camera's trajectory. We are using RANSAC to filter out inliers for mapping from feature data set.Now image based 3-D mapping is possible as mentioned by S. Agarawal et. al in [4]. This technique can be used to model 3-D map using colour and depth information of image. But as for indoor environment light conditions and texture less wall may cause problem.

As application require high data computation, we used multiple cores to parallelize it with OpenMP. We compare our parallel algorithm results with serial implementation of algorithm.

Applications:These 3-D maps can be used in gaming, in robotics for robot navigation and object detection. 3-D model of indoor environment can be added to map, to get more information about internal infrastructure.

II. Related Work

A. 3-D Localization

Geometrically modelling of physical environment is long time done manually by specialists using specialized tools. Automatically mapping an environment is challenging. Dieter Fox et al.[5] proposed a algorithm for effective position

estimation of mobile robot which is using a laser scanner to build map. Robot mapping has been shown to be robust, but is mostly limited to 2D maps and relies on expensive hardware. Jeffrey Hightower et al. [6] gave much more attention over problems in location systems in the Ubiquitous Computing, A variety of signals has been used in indoor localization, such as 802.11, GSM , and recently power line signals . There is a limit on the localization accuracy using these low-rate signals, the state of the art being around 0:5 meters.

B. 3-D Modeling

Recently Noah Snavely et al. [7] showed, image-based 3D modelling is feasible. Photo Tourism being a prominent example how 3D structures can be recovered by analysing and matching photos. Sameer Agrawal's et al. [4] research work shows promising results on city-scale outdoor scenes. Vision-based 3D modelling techniques have been gaining popularity in recent years. Building on multi-view geometry and in particular bundle adjustment algorithms, 3D structures can be recovered from a set of 2D views. PhotoTourism designed by Noah Snavely et al. [7] is an example where sparse 3D models are constructed from web photos.

There have been successful efforts to build real-time systems for 3D structure recovery. Davison et al. [8] built real-time SLAM (simultaneous localization and mapping) systems using monocular cameras. Pollefeys et al. [9] proposed real-time solutions for street-view reconstruction using GPS and video data. Many real-time systems are limited to small-scale spaces. Xiaodong Li et al. [10] designed system which using SIFT and SURF for feature extraction and RANSAC for navigation based on digital camera image. H. Du et al. [2] design system which is using interactive user input to design 3D model.

III. System Architecture

Figure 1 is showing system architecture. With the help of depth camera, data is collected. Input consist of colour plus depth information. As each input frame consist of around 300K points so for further processing, input is filtered and key features are extracted from it. Here we are using Scale Invariant Feature Transformation(SIFT) as keypoints of input cloud for further processing. Using visual odometry local alignment between two consecutive frames is calculated. From this local alignments global position of each frame in model is calculated. Each frame is registered to its global position and model is updated. This 3-D model can be visualised by available 3-D visualizer.

A. Simultaneous Localization And Mapping

While preparing 3D map it is important to localize the current camera location with respect to 3D co-ordinates. SLAM is technique which is used generate map of unknown location or update the map of known location.In SLAM technique, it continuously captures data, by comparing this data with consecutive frames it localize current frame. Simultaneously it register this frame to global model to generate complete map.

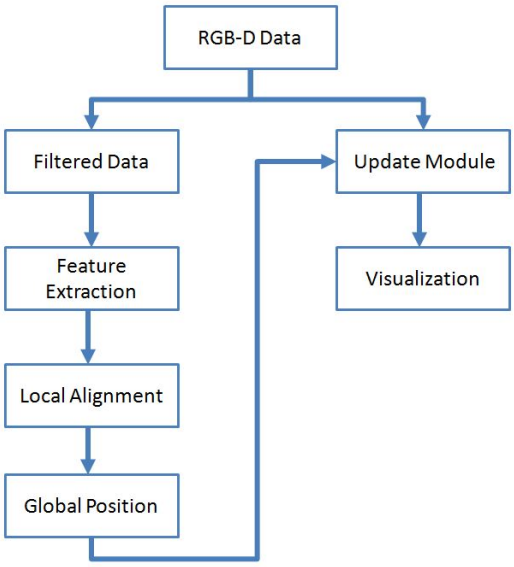

Figure 1. Block diagram of System Architecture

IV. Feature Extraction

Depending on the number of points generated by the kinect, it might make sense to use only a few selected points to calculate the optimal transformation between two point clouds, and then apply this transformation on all points. Depending on the source of the data, it also turns out that some points are more suitable than others as it is easier to identify matches for them.
In case of RGB data, where SIFT features are more important, while in case of planar objects with grooves normals to the surface are more important features.

A. Surface Normals

Surface normals are important properties of a geometric surface, and are heavily used in many areas such as computer graphics applications, to apply the correct light sources that generate shadings and other visual effects.

B. SIFT Key points

SIFT image features provide a set of features of an object that are not affected by many of the complications experienced in other methods, such as object scaling and rotation. While allowing for an object to be recognized in a larger image SIFT image features also allow for objects in multiple images of the same location, taken from different positions within the environment, to be recognized. SIFT features are also very resilient to the effects of noise in the

image. The SIFT approach, for image feature generation, takes an image and transforms it into a large collection of local feature vectors. Each of these feature vectors is invariant to any scaling, rotation or translation of the image. This approach shares many features with neuron responses in primate vision.

C. Adaptation of SIFT Keypoints for 3D

The (r,g,b) coordinates of the point clouds are converted to an intensity value i, so now p = (x, y, z, i). The modification of the algorithm is applied in the scale-space extrema detection step. In the original algorithm, the key points are identified as local minima/maxima of the Difference-of-Gaussians (DoG) images across scales. This is done by comparing each pixel in the DoG images to its eight neighbours at the same scale and nine corresponding neighbouring pixels in each of the neighbouring scales. If the pixel value is the maximum or minimum among all compared pixels, it is selected as a candidate key point. In point clouds a 3D point is used as the pixel and its neighbouring pixels are the nearest neighbours in (x, y, z) coordinates obtained using approximate nearest neighbour search within a fixed radius set by the scale in a kd-tree representation of the point cloud.

That traditional 2D SIFT key points are computed by repeatedly blurring an image with Gaussian filters of increasing scale, subtracting the different scales to get a difference-of-Gaussian (DoG) scale space, and then finding local minima and maxima in that DoG scale space. So we can do the same thing in a 3D point cloud as long as we have a way of performing Gaussian blurs and searching for local extrema.

V. Pair Alignment

While collecting data, we scanned the area by moving depth camera inside room. Microsoft Kinect collect data at the rate of 30FPS , so to keep some noticeable difference between each frame we captured every fifth frame. We may consider any nth frame depend on speed of movement of camera provided that there must be more overlapping between two consecutive frames. Now each frame is slightly different and more overlapped with its previous frame. We have to find out the exact Transformation i.e. Rotation and Translation of that frame with respect to previous frame.

Figure 2 is showing results of pair alignment. In first part inside red circles there are duplicated objects i.e. Notebook and Box. This is because there are two frames captured from slightly different locations and overlapped together without any transformation. We can see inside green circles, after the alignment of two frames, two objects are aligned properly.

Iterative Closest Point

To localize new arriving frame it is aligned with its just previous frame. This alignment is in terms of transformation

Figure 2. Pair alignment output

matrix (consist of rotation and translation matrix). Here Iterative Closest Point[11] algorithm is used to find out alignment. ICP is basically used to find the best transformation that minimizes the distance between two point clouds.

ICP is a registration technique that uses geometry information (X, Y, Z) and not intensity/color to register the source point cloud to the target point cloud. It works iteratively to find out best solution.

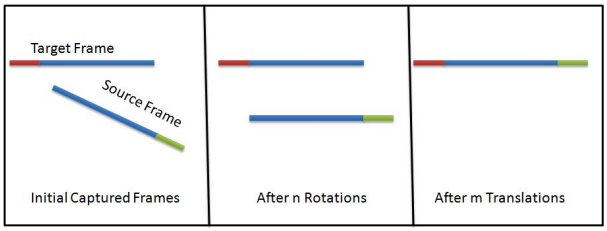

Figure 3. Working of ICP Algorithm

Figure 3 is giving idea about working of ICP. Two lines are considered as point clouds. The blue part is showing overlapped part in two consecutive frames. Initially the source cloud is completely misaligned with respect to the target cloud, after n iterations the source cloud seems to be aligned in terms of rotation. Next, after a few more iterations (n + m), the source is now completely aligned to the target cloud in terms of Rotation and Translation. Source cloud is overlapped with target cloud. ICP allows us to converge a source point cloud having 6 degrees of freedom (DoF) from the target point cloud.

A. Euclidean distance

To define which is closest point on the target cloud , the point which has the smallest Euclidean distance from a point on the source cloud is considered. If we had two points p (on the target cloud) and q (on the source cloud), the Euclidean distance would be the line segment connecting point p and q. The distance can be quantitatively calculated by the below formula . For point P and Q, in our point cloud (3D) case we have P (x,y,z) and Q (x,y,z):

p1 = x co-ordinate — q1 = x co-ordinate
p2 = y co-ordinate — q2 = y co-ordinate
pn = z co-ordinate — qn = z co-ordinate

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad (1)$$

Each ICP Iteration consist of the following steps:

1) Find Closest Points: Using the k-d search tree find out the Euclidean distance between the target/source point clouds.
2) Calculate Alignment: Calculate what is the best rotation/translation required to be performed on the source point cloud to align it with the target point cloud.
3) Regenerate Source Cloud: After applying the transformation (rotation and/or translation) to the source cloud, regenerate the modified point cloud. The visual effect of this is the source cloud either appears closer or further from the target cloud.
   The transformation matrix is composed of an inner 3*3 rotation matrix and the 4th column is the translation vector (3*1).

$$TransformationMatrix = \begin{vmatrix} R1 & R2 & R3 & x \\ R4 & R5 & R6 & y \\ R7 & R8 & R9 & z \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

B. Local Alignment

Local alignment is frame to frame registration step. So by using (n-1) th cloud as source and (n) th cloud as target the transformation matrix is calculated. With this local transformation matrix, source cloud is transformed to align properly with target cloud. This local alignment is refer to alignment with respect to only consecutive frame.

C. Global Alignment

To prepare one consistent model, every frame should be transformed with respect to one frame i.e. first frame.With local alignment we are getting alignment with respect to consecutive frame. This local alignment can be used to find out global alignment.

Initial Transformation Guess

The ICP method has seen many improvements from its original form, from using non-linear optimization methods, finding good initial guesses, or estimating better point features, to addressing the problem of ICP's computational complexity. Initial guess for ICP is calculated using RANSAC inliers found by FPFH descriptors [12].

D. FPFH Descriptor

Fast Point Feature Histograms (FPFH) are informative pose-invariant local features which represent the underlying surface model properties at a point p. Their computation is based on the combination of certain geometrical relations between p's nearest k neighbours. Descriptors are determined over the points where SIFT keypoints are found. So now these descriptors giving more information calculated based on neighbours in the form of histogram. The FPFH features [13] are pose invariant and their discriminative power makes them good candidates for point correspondence search in 3D registration.

E. Correspondence Estimation

Correspondence grouping algorithms cluster the set of point-to-point correspondences obtained after the 3D descriptor matching stage. Using correspondence algorithm we are finding set of points in target frame which are matching with source frame descriptors.

1) Correspondence Using SIFT points We can feed SIFT points obtained from frames to find out correspondence based on x,y,z co-ordinates matching.
2) Correspondence Using FPFH Descriptors FPFH descriptors are feed to correspondence finding algorithm. It compare the histograms of the key points and determine the correspondence.

F. Finding Inliers and Initial Transformation Guess

Most of the obtained correspondences are wrong, which are negatively affecting registration results. So to remove wrong correspondences and find out inliers which are fit into appropriate geometrical model RANSAC algorithm [10] is used. With these inliers the transformation matrix required to map source cloud on target cloud is calculated. This transformation matrix is used as initial guess for ICP.

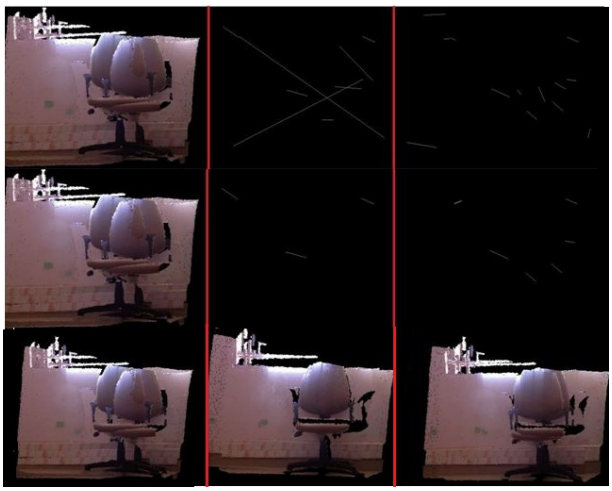

Figure 4. Comparison of methods of finding correspondence between two clouds using FPFH descriptor and using SIFT keypoints.

In Figure 4, left part of first row is showing two cloud frames captured at different time from different locations so not aligned, middle part is showing correspondence pairs fond using FPFH and right part is showing correspondence pairs found using only SIFT points. In row two, middle and right part are showing respective inliers after applying RANSAC. In row three, middle and right part, point clouds are shown after applying transformation obtained from respective methods. We can see alignment using FPFH is better than SIFT points.

G. Point Density due to Redundant Data

As to get better results from ICP we have to capture frames in such way that they are mostly overlapped but after getting transformation matrix for each frame if we transform every frame into result we will get very dense cloud and in which most of the part will be drawn again. To solve this problem we are using transformation matrix. If the translation of any frame is more than certain limit then we are transforming that frame in to resultant cloud. Here 10cm as limit is used.

Parallel Modules

As serial version of algorithm require high computing time because of huge data computation, it is possible to use multiple cores for data parallelism. OpenMP is used for data parallelism. Different modules without any dependency are created.

Following modules are parallelize using OpenMP

1) Cloud filtering
2) Computing normals
3) Computing SIFT key points
4) Computing FPFH descriptor
5) Finding correspondences, inliers and initial guess
6) Finding transformation between consecutive frames
7) Transforming point cloud

Captured input consist of n frames then each frame can be given to each core to run these modules. So as many cores are there those many frames will be processed together. With this approach running time is considerably reduced.

VI. Results

Figure 5 shows the result after registration of 40 captured frames with constantly moving kinect.Here we can see it has detected proper position of each frame and registered at its location. Edge of the two walls is also aligned.

Frame registration module consist of cloud filtering, finding sift points, finding local alignment between pair of clouds and finding global alignment for each cloud. These steps are independent of each other so we can parallelize them on multi cores. The i7 system with, 8 cores processor is used.Table I showing Time required for Serial and Parallel implementation of Frame Registration Module using OpenMP on 8 core machine. By looking at values we can see that parallel algorithm took much less time compare to serial algorithm. We have calculated Speed Up factor and algorithm Efficiency using following formula.

Figure 5. Snapshot of 3D model captured in the lab.

$$Speedupfactor = SerialTime/ParallelTime \quad (2)$$

$$Efficiency = Speedupfactor/NumberOfCores \quad (3)$$

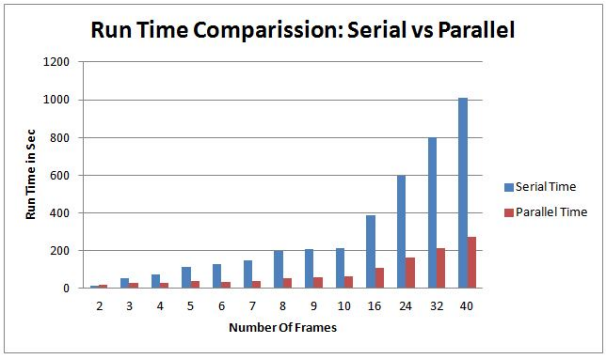

Figure 6. Graph showing running time comparison between Serial and Parallel version of frame registration module.

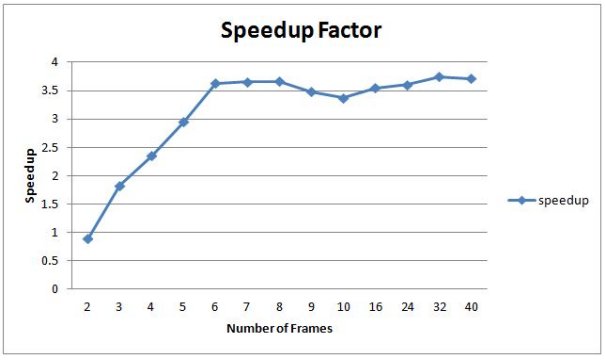

Figure 7. Graph showing speed up factor of parallel registration module.

Table I
RUNTIME COMPARISON OF SERIAL AND PARALLEL MODULE

# Frames	Serial Time(s)	Parallel Time (s)	Speedup Factor	Efficiency(%)
2	16	18	0.89	11.11
3	53	29	1.83	22.84
4	73	31	2.35	29.44
5	112	38	2.95	36.84
6	127	35	3.63	45.36
7	150	41	3.66	45.73
8	198	54	3.67	45.83
9	209	60	3.48	43.54
10	216	64	3.38	42.19
16	387	109	3.55	44.38
24	595	165	3.61	45.08
32	802	214	3.75	46.85
40	1009	272	3.71	46.37

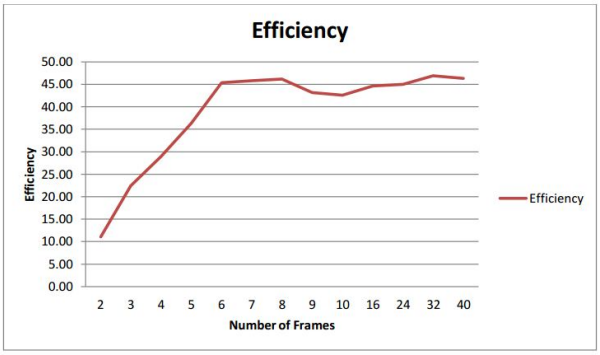

Figure 8. Graph showing Efficiency of parallel registration module.

Graphs 6, 7, 8 are showing that as we are increasing the numbers of frame we are getting better results. But due to interdependencies between modules we can't achieve still more higher speed up factor. Here we achieve speed up factor up to 3.7. The efficiency of parallel algorithm is up to 47 percent.

VII. CONCLUSIONS

Use of surface SIFT points as key point for alignment gives better results as comparative to normal key points. Inliers found using FPFH descriptors are more accurate than inliers found using just SIFT points. Initial transformation guess found using RANSAC is improving accuracy but small wrong guess may leads to bad results. Results are showing that, due to use of OpenMP for parallelism on 8 cores gives speedup upto 3.75 and around 47% parallel algorithm efficiency.

REFERENCES

[1] "Microsoft kinect," http://www.xbox.com/en-IN/Kinect.

[2] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox, "Interactive 3d modeling of indoor environments with a consumer depth camera," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 75–84.

[3] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Manhattan-world stereo," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1422–1429.

[4] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 72–79.

[5] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proceedings of the National Conference on Artificial Intelligence*. JOHN WILEY & SONS LTD, 1999, pp. 343–349.

[6] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, 2001.

[7] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," in *ACM transactions on graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 835–846.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, 2007.

[9] M. Pollefeys, D. Nistér, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S.-J. Kim, P. Merrell *et al.*, "Detailed real-time urban 3d reconstruction from video," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

[10] X. Li, N. Aouf, and A. Nemra, "3d mapping based vslam for uavs," in *Control & Automation (MED), 2012 20th Mediterranean Conference on*. IEEE, 2012, pp. 348–352.

[11] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp."

[12] "Point cloud library," http://pointclouds.org.

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