Consumer RGB-D Cameras and their Applications

Krystof Litomisky

klitomis@cs.ucr.edu

University of California, Riverside

Spring 2012

Introduction

In November 2010, Microsoft released the Kinect RGB-D sensor as a new Natural User Interface (NUI) for its XBOX 360 gaming platform. The Kinect, like other RGB-D sensors, provides color information as well as the estimated depth for each pixel. At $150, the Kinect is an order of magnitude cheaper than similar sensors that had existed before it. This has dramatically reinvigorated interest in RGB-D sensors and their applications in areas such as Natural User Interfaces, reconstruction and virtual reality, or 3D mapping.

The purpose of this document is to provide an overview of currently-available consumer RGB-D sensors, applications for which these sensors are being used, and various supporting frameworks (such as libraries and SDKs).

The main focus of this document is on the applications of RGB-D sensors to three-dimensional mapping.

Table of Contents

1. Consumer RGB-D Sensors

RGB-D sensors combine RGB color information with per-pixel depth information. Sensors that provide such data have existed for years, including the Swiss Ranger SR4000 and PMD Tech products (see http://www.hizook.com/blog/2010/03/28/low-cost-depth-cameras-aka-ranging-cameras-or-rgb-d-cameras-emerge-2010). However, these sensors cost around $10,000 each. By contrast, new consumer RGB-D sensors cost less than $200.

The per-pixel depth sensing technology that is used in consumer RGB-D cameras was developed by PrimeSense (http://www.primesense.com/en/technology/115-the-primesense-3d-sensing-solution). The technology is patented (United States Patent US7433024).

The technology is licensed for use in the commercially-available Microsoft Kinect and Asus Xtion PRO sensors. Both were created as consumer products for NUI applications. In particular, the Microsoft Kinect is an Xbox accessory, and became the fastest-selling consumer electronics device in the world after its launch in November 2010 (see http://community.guinnessworldrecords.com/_Kinect-Confirmed-As-Fastest-Selling-Consumer-Electronics-Device/blog/3376939/7691.html).

1.1. Technology

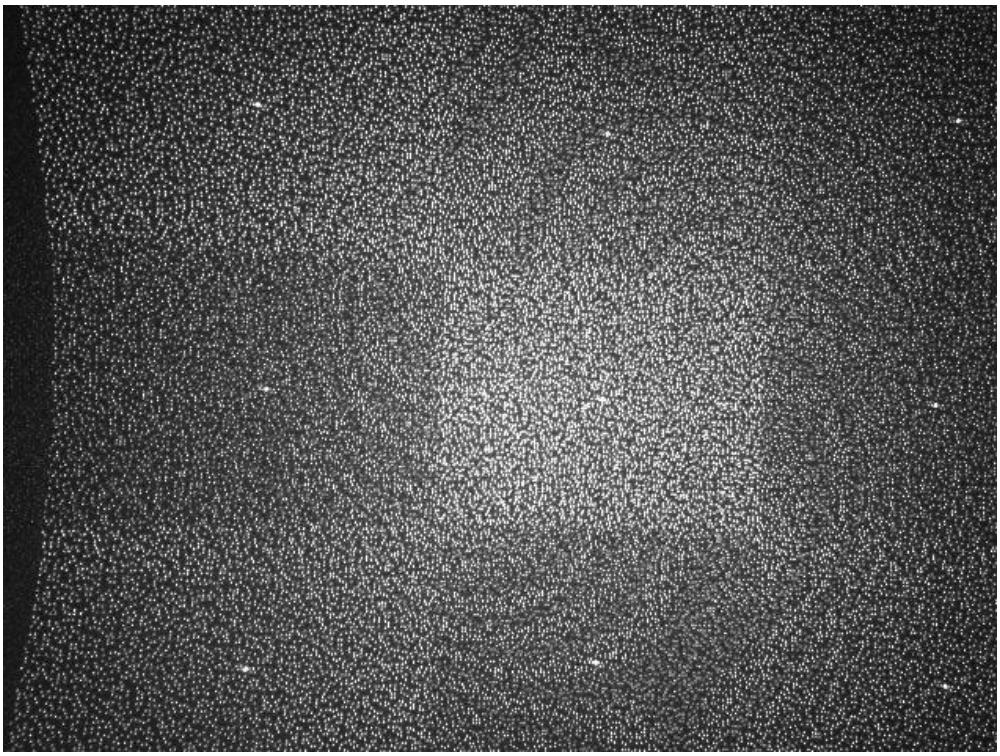

Figure 1: MS Kinect infrared pattern. Source: http://www.ros.org/wiki/kinect_calibration/technical.

The PrimeSense sensor projects an infrared speckle pattern (Figure 1). The projected pattern is then captured by an infrared camera in the sensor, and compared part-by-part to reference patterns stored in the device. These patterns were captured previously at known depths. The sensor then estimates the per-pixel depth based on which reference patterns the projected pattern matches best [REF: primesense patent: http://www.google.com/patents/US20070216894?dq=primesense].

The depth data provided by the infrared sensor is then correlated to a calibrated RGB camera. This yields an RGB image with a depth associated with each pixel. A popular unified representation of this data is a *point cloud*: a collection of points in three dimensional space, where each point can have additional features associated with it. With an RGB-D sensor, the color can be one such feature. Additionally, approximated surface normals are also often stored with each point in a point cloud.

1.2. Devices

Because consumer RGB-D sensors were created as NUI devices, they have certain characteristics that limit their utility for mapping applications. In particular, the field of view is much smaller than mapping-specialized sensors, and the depth resolution deteriorates notably with depth (section 1.3).

Microsoft Kinect

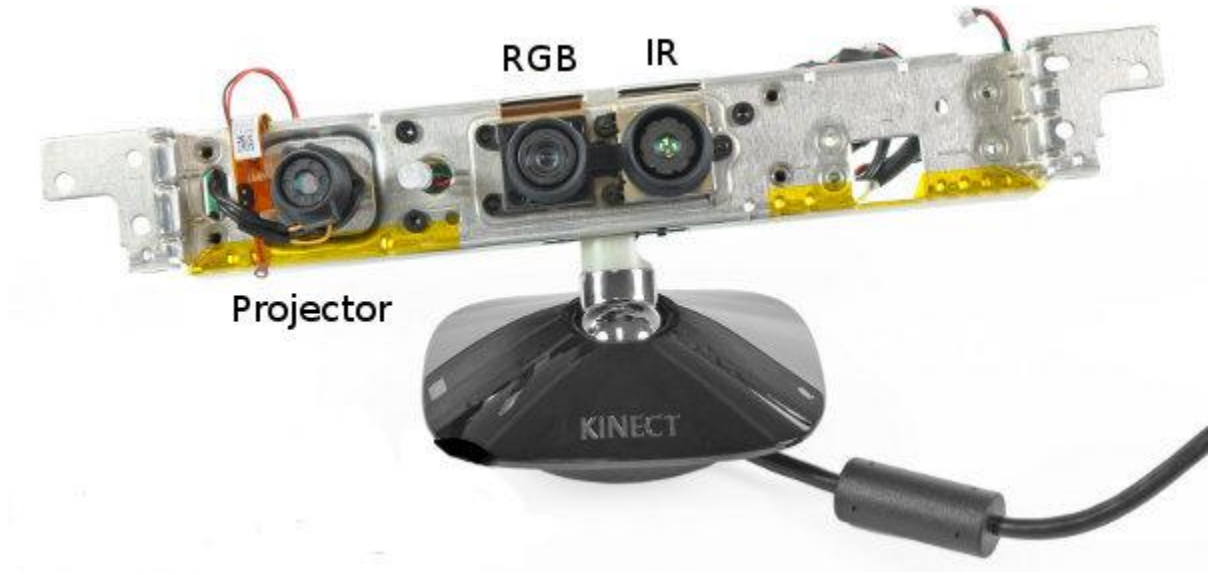

Figure 2: The Microsoft Kinect with the cover taken off. Source: http://www.ros.org/wiki/kinect_calibration/technical

The standard version of the Microsoft Kinect sensor is commercially available for around $150. In addition to the RGB and depth sensors, the Kinect also has a microphone array, which is particularly helpful in NUI applications. The technical specifications of the Kinect sensor are in Table 1.

Field of view	43° vertical by 57° horizontal field of view
Frame rate (depth and color stream)	30 frames per second (FPS)
Default resolution, depth stream	VGA (640 x 480)
Default resolution, color stream	VGA (640 x 480)
Audio format	16-kHz, 16-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing, such as acoustic echo cancellation and noise suppression

Table 1: Microsoft Kinect technical specifications. Source: [REF: http://msdn.microsoft.com/en-us/library/hh855355.aspx]

Asus Xtion PRO LIVE

Figure 3: Asus Xtion PRO LIVE. Source: http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE/.

The Xtion sensor should retail for less than $200; however, as of May 2012, I have been unable to find a retailer that sells the Xtion, and it seems that it may be discontinued. Asus has an Xtion PRO version, which only provides depth data, and an Xtion PRO LIVE version, which provides depth, color, and audio (using a microphone array, like the Kinect). Table 2 shows the technical specifications of the Xtion PRO LIVE sensor.

Field of View	58° H, 45° V, 70° D
Depth Image Size	VGA (640x480) : 30 fps QVGA (320x240): 60 fps
Resolution	SXGA (1280*1024)
Interface	USB2.0

Table 2: Asus Xtion PRO LIVE technical specifications. Source: http://www.asus.com/Multimedia/Motion_Sensor/Xtion_PRO_LIVE/#specifications.

The Leap

The Leap sensor is much smaller than either of the above RGB-D sensors, and is designed for much finer NUI on a smaller scale. It is not available yet, but Leap Motion, the company behind The Leap, is accepting preorders. For more information, see http://www.leapmotion.com/.

1.3. Depth Resolution

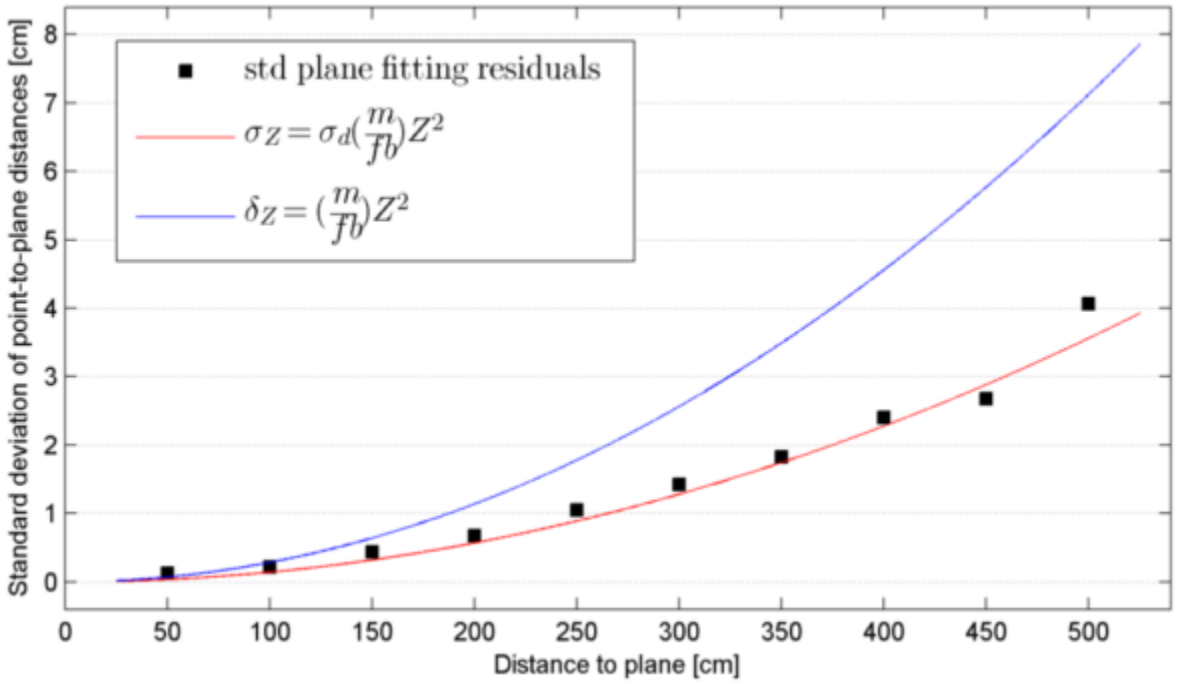

Figure 4: Microsoft Kinect depth resolution vs. depth (Khoshelham & Elberink 2012). Theoretical random error is shown in red (bottom curve), while the theoretical resolution is shown in blue (top curve).

The accuracy of the depth data provided by consumer RGB-D sensors deteriorates as the objects in the scene get further away from the sensor. This is not a significant issue for NUI applications, where the user can stay at a range that is suitable for the sensor, but it is important for mapping applications.

Khoshelham and Elberink looked at how the depth resolution of data from the Microsoft Kinect changes as objects get further from the sensor (Khoshelham & Elberink 2012). In particular, they pointed the sensor at a door that was a known distance away, and looked at the standard deviations of the distances of points provided by the sensor from the true plane. Figure 4 summarizes their results.

My own previous experiments confirm this. Figure 5 shows the depth data with approximated surface normals of a mannequin captured at different distances. As in the experiments of Khoshelham and Elberink, the depth resolution gets dramatically worse as the object being observed gets further from the sensor.

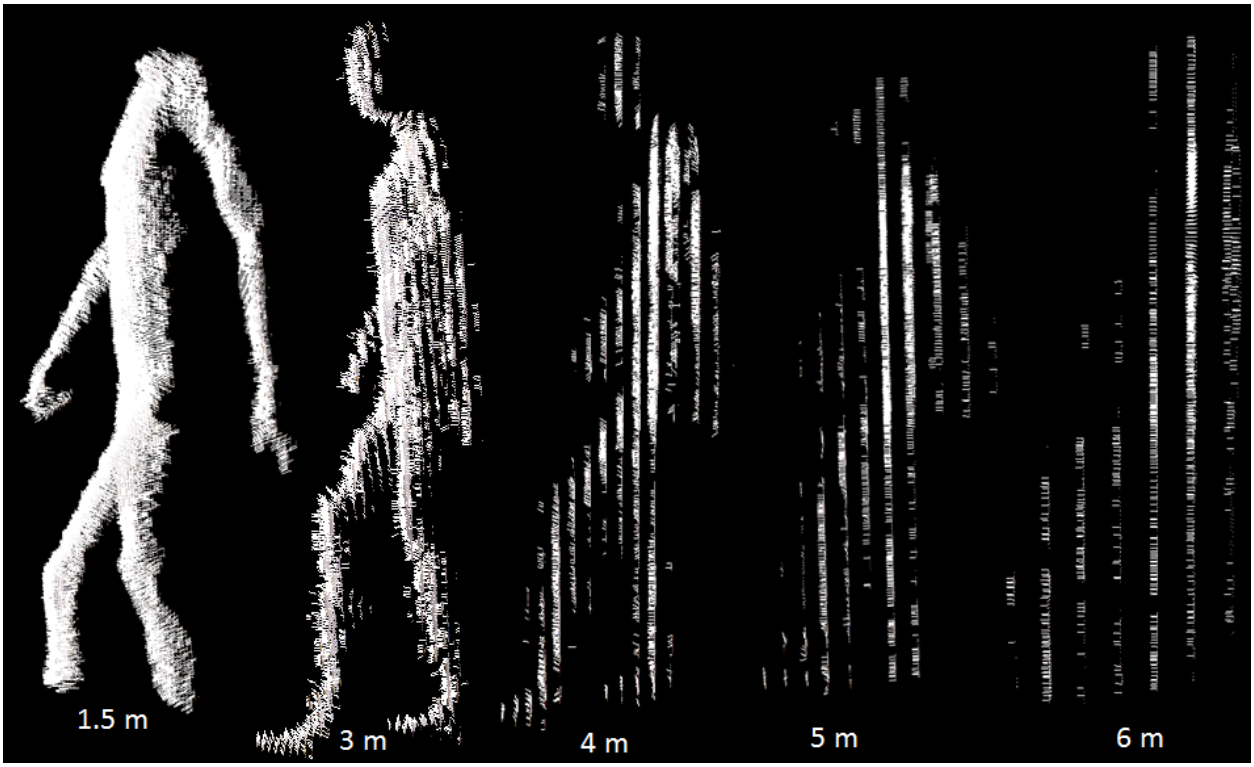

Figure 5: Depth data of a mannequin captured from the front and shown from the side. From left to right, the mannequin is shown as captured at 1.5 m, 3 m, 4 m, 5 m, and 6 m.

Lange et al. compare the quality of the depth data provided by the Kinect to some commercial sensors designed specifically for robotics applications (Lange et al. 2012). A summary of their results is in Table 3. The most noteworthy observation regarding this data is the significantly higher standard deviation of repeated measurements at known distances, particularly as the distance from the sensor increases. At a distance of 4 meters, the Kinect had a standard deviation of 27.5mm, compared to just 6mm for the

SwissRanger 4000 and 4mm for the PMD CamCube 3.0. Nevertheless, the authors are convinced that the Kinect's low price will make it an attractive option.

Sensor	max. range	resolution	field of view in deg	repeatability in mm (1σ)	weight
Kinect	10 m	640×480	57.8×43.3	7.6 @ 2 m, 27.5 @ 4 m	440 g
SwissRanger 4000	8 m	176×144	43×34 / 69×56	4 / 6	470 g
PMD CamCube 3.0	7 m	200×200	40×40	3 @ 4 m	1438 g

Table 3: comparison of depth data quality of different sensors (Lange et al. 2012).

2. 3D Mapping and Control Applications

Intro paragraph to using consumer RGB-D sensors for SLAM: limitations due to field of view, max distance from sensor, quantization, huge amount of data

RGB-D Mapping

Henry et al. got access to a prototype of an RGB-D sensor directly from PrimeSense before the sensor became commercially available; this led to work published in the International Symposium on Experimental Robotics (ISER) (Henry et al. 2010). The paper was later extended and published in the International Journal of Robotics Research (Henry et al. 2012).

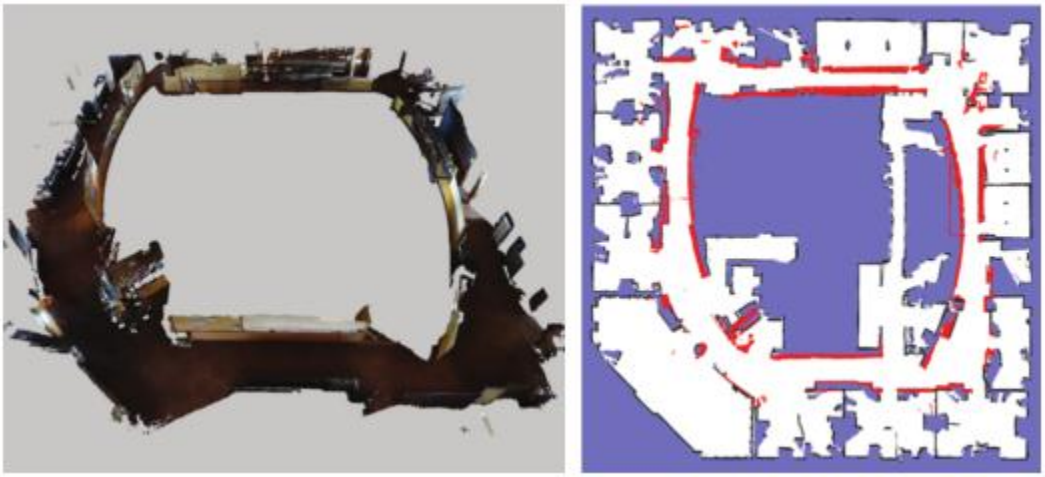

Figure 6: A map created with Henry et al.'s RGB-D Mapping framework (left), and the corresponding ground truth (right) (Henry et al. 2012).

Henry et al. first perform a pairwise rough initial alignment using SIFT visual features (Lowe 1999) and a Random Sample Consensus (RANSAC) algorithm, which is then refined using the depth-based Iterative Closest Point algorithm (ICP) (Besl & McKay 1992). To deal with the accumulation of alignment error, the

RGB-D mapping framework does keyframe-based loop-closure detection and global constraint optimization. Loop closure is identified using SIFT features.

The RGB-D mapping framework deals with the huge amount of data produced by storing the maps not as point clouds, but rather as surfels. Surfels are small planar patches with an associated color, location, surface orientation, and size (Pfister et al. 2000).

Figure 7: comparison of data in a point cloud representation (left, 56.6 million points) and the surfel representation used by the RGB-D Mapping framework (right, 2.2 million surfels) (Henry et al. 2010).

According to the paper, the surfel representation shrinks the size of the data by about a factor of 32. In addition, "because surfels have a notion of size … we can reason about occlusion, so if an existing surfel is seen through too often, it can be removed" (Henry et al. 2010). Unfortunately, the authors do not include any experiments with moving objects, which would be an interesting test of how well the surfel-based approach can work.

Interactive 3D Modeling of Indoor Environments

Du et al. published an extension of the RGB-D Mapping framework described above in UbiComp '11 (Du et al. 2011). Like the RGB-D Mapping framework, the frame-by-frame alignment is done using RANSAC with SIFT visual features, combined with a visibility conflict term (how likely it is that a transformation is correct, based on where the seen points would lie in the merged cloud). To deal with the depth resolution issues, the authors use depth data only up to 5 meters. The system runs at about 3 frames per second.

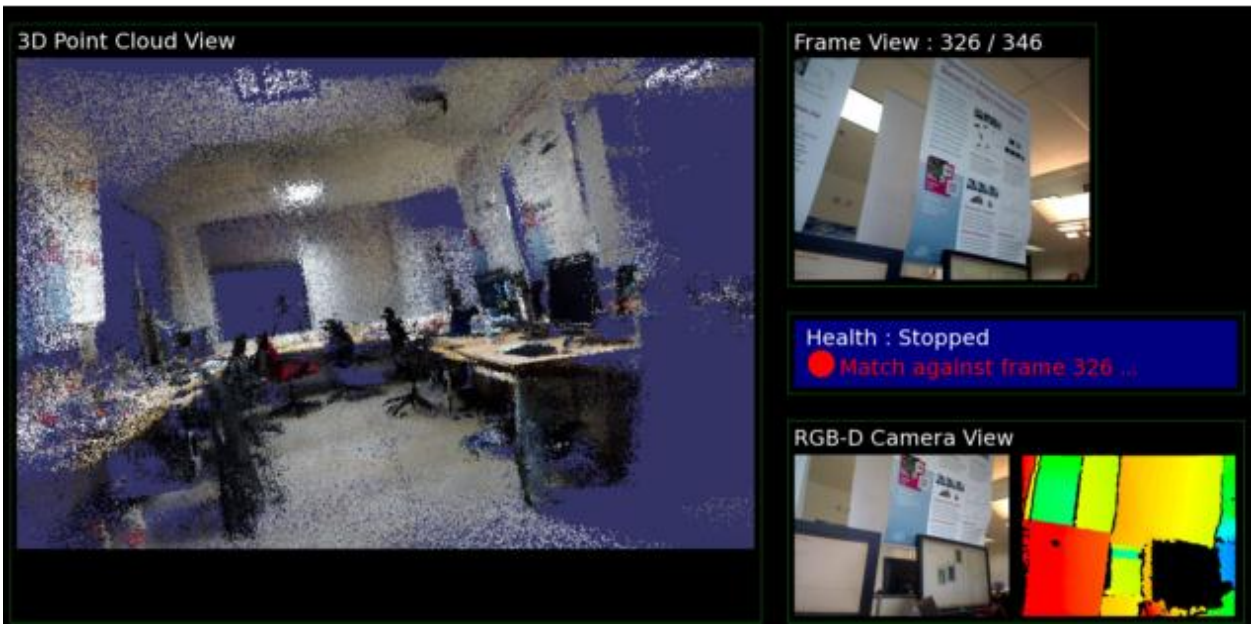

Figure 8: Interactive 3D Modeling of Indoor Environments system screenshot (Du et al. 2011).

The key difference between the RGB-D Mapping framework and this work is that this work leverages user feedback to deal with registration issues, solve loop-closure dilemmas, and encourage more complete exploration of the environment.

The authors do not discuss issues that may arise from the presence of moving objects and how their work might deal with them.

Autonomous Flight

Huang et al. present an approach to use an RGB-D sensor in an autonomous micro air vehicle (MAV) both for navigation and mapping in a paper published in 2011 in the International Symposium on Robotics Research (Huang et al. 2011). In order to achieve real-time rates, the authors decouple local position estimation (which is quickly done onboard the MAV – about 25ms per frame) from SLAM computations (which are slower, and done offboard on a laptop). However, information from the SLAM computations periodically refines the local position estimates.

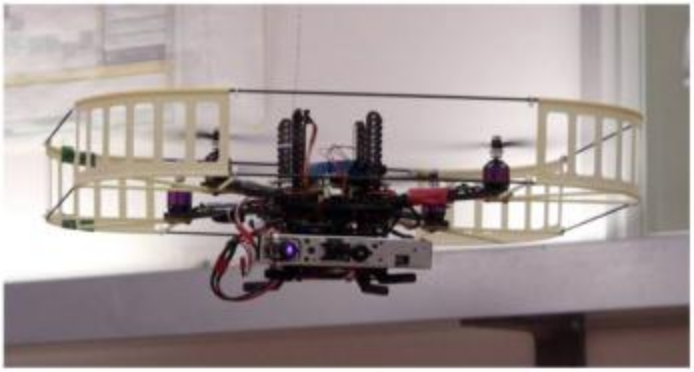

Figure 9: Autonomous micro air vehicle (MAV) used with an RGB-D sensor mounted at the front (Huang et al. 2011).

To extract visual odometry information, the authors extract features using the FAST corner detector (Rosten & Drummond 2006) at each level of a Gaussian pyramid of each image frame. The authors then estimate frame-to-frame rotation by minimizing the sum of pixel errors between downsampled versions of consecutive frames. This estimate is used to constrain the search space for feature matches between the two frames. The authors then use a greedy max-clique graph algorithm (rather than RANSAC) to identify feature match inliers, which are then used to get a final estimation of the frame-to-frame transformation.

The raw RGB-D data is transmitted from the MAV to a laptop, which then detects loop closures (and performs appropriate global constraint optimizations), and creates a downsampled 3D map of the environment. Any pose corrections are transmitted back to the MAV along with a current voxel map of the environment.

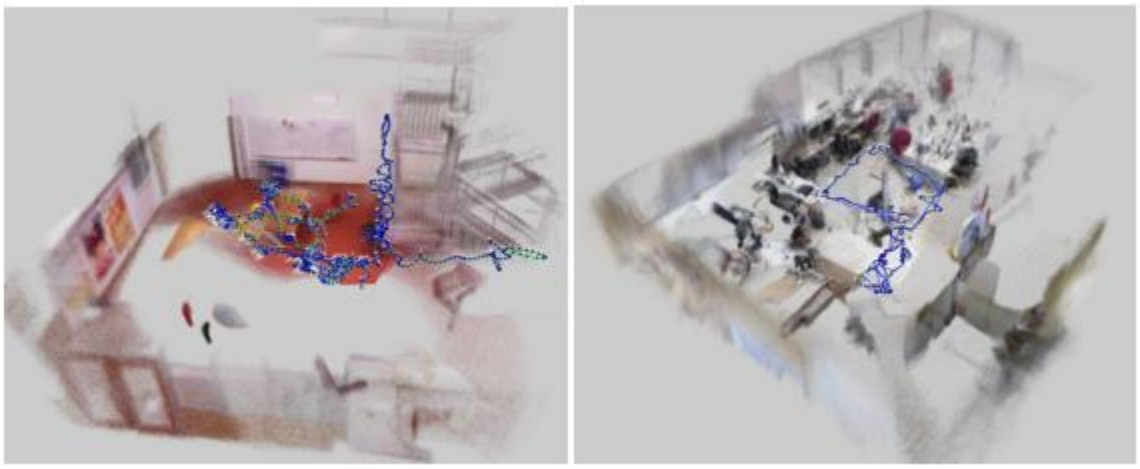

Figure 10: Point cloud maps created by the algorithm, with the trajectories of the MAV shown (Huang et al. 2011).

The authors performed quantitative experiments in a 11m x 7m x 4m room with varying degrees of clutter and easy-to-extract visual features, presenting a comparison of different methods for performing the computations at each stage of the algorithm. Due to issues such as motion blur or a lack of good features, the authors state that their algorithm "is unlikely to have been capable of autonomously flying the MAV through the entire recorded trajectory". The authors state that their algorithm performs significantly better in feature-rich environments, however.

The algorithm assumes a completely static environment and "relatively slow" motion of the MAV. Regarding moving objects, the authors state that "relaxing the static environment assumptions will likely require better ways of detecting the set of features useful for motion estimation. When moving objects comprise a substantial portion of the visible image, the maximal clique of consistent feature matches may not correspond to the static environment."

Lange et al. also used the Kinect sensor to guide an MAV, focusing on particularly confined, GPS-denied areas such as corridors (Lange et al. 2012). The authors implemented all controllers on a custom microcontroller board to ensure real-time rates. The MAV also includes a sonar sensor that monitors the quadrotor's distance from the ground, and a camera that monitors the velocity.

The first step of Lange et al.'s point cloud processing pipeline is to downsample the clouds dramatically: from around 300,000 points to around 3,000. The purpose of this is to speed up processing. After downsampling, the authors extract large planes using a sample consensus approach (MLESAC) provided by the Point Cloud Library (Rusu & Cousins 2011). Having done this, the authors calculate the distance of the quadrotor to each plane, as well as the roll, pitch, and yaw angles. These authors then assign each plane a label (floor, ceiling, left, right, or front) based on these angles. Based on the plane labels and parameters, the authors calculate the desired motion commands to keep the MAV moving along the corridor.

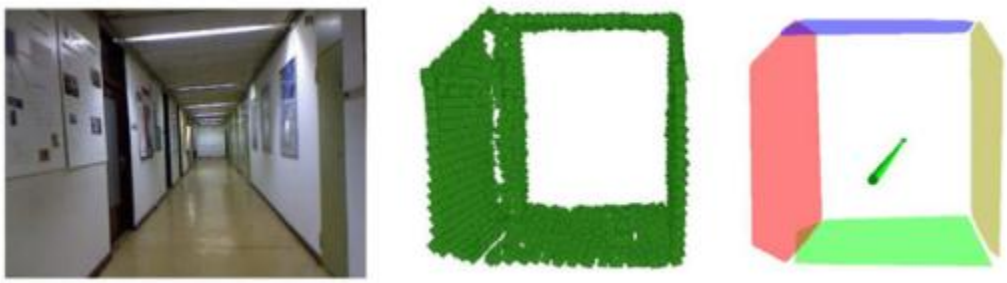

Figure 11: Plane estimations from Lange et al.'s MAV (Lange et al. 2012).

Realtime Visual and Point Cloud SLAM

Fioraio and Konolige focus on improving the runtime of frame-to-frame registration, with the goal of more fully using RGB-D cameras' high framerate (Fioraio & Konolige 2011). Towards this end, the authors develop a SLAM algorithm that runs ICP on downsampled depth frames, can integrate additional features such as color information, and optimizes frame-to-frame transformations with respect to global constraints. The authors do not, however, fully integrate loop-closure and global optimization on visual feature constraints, stating that this work is ongoing.

Figure 12: Scene reconstructed and optimized from 72 different views (Fioraio & Konolige 2011).

For frame-to-frame registration, the authors use a RANSAC-computed transformation based on visual features as the initial transformation estimate for the Iterative Closest Point (ICP) algorithm. Then, the authors uniformly downsample both frames, estimate normals, and then look for a match between the two frames. A match is kept if the distance between the points is small enough, the difference between the normal angles is small enough, and the match does not belong to too many other corresponding pairs. The authors then minimize the Euclidean distance between all matched pairs using point-to-point, point-to-plane, or plane-to-plane error metrics.

The authors observe that using visual features significantly improves alignment performance. In particular, the authors use features detected with FAST (Rosten & Drummond 2006) and extracted with BRIEF (Calonder et al. 2010). BRIEF is a binary-string-based point descriptor; its main strength is that

descriptor similarity can be evaluated using the Hamming distance, which is much faster than computing the L2 norm.

Towards a Benchmark for RGB-D SLAM

In response to the great interest in using consumer RGB-D cameras for SLAM, Sturm et al. have published a dataset to provide a benchmark for the evaluation of SLAM systems that use RGB-D data (Sturm et al. 2011). The dataset contains sequences of RGB-D frames collected at 30 FPS and 640x480 resolution, with an associated ground-truth trajectory of the sensor, which was captured with a motion-capture system.

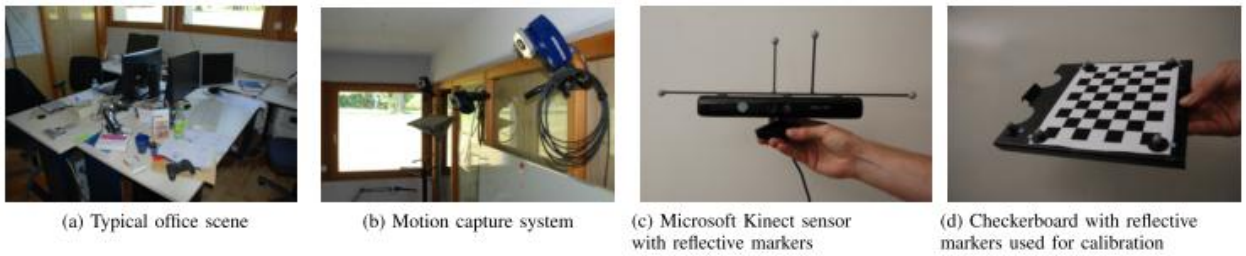

Figure 13: The dataset collection setup of Sturm et al. RGB-D SLAM benchmark (Sturm et al. 2011).

The dataset consists of sequences collected in a typical office setting, with varying movement speeds and environment sizes. In addition, the authors also include sensor accelerometer data. The final dataset contains nine sequences, and totals about 50 GB. It is available under the Creative Commons License from https://cvpr.in.tum.de/research/ datasets/rgbd-dataset.

In addition to the dataset, the authors also propose an evaluation metric, based on the difference between the true and estimated motion of the sensor with respect to the stationary motion capture system. At present, however, there is no agreed-upon, objective way to evaluate the quality of maps reconstructed by SLAM systems.

3. Reconstruction and Virtual Reality

RGB-D sensors can be used on a much smaller scale than SLAM to create more detailed, volumetric reconstructions of objects and smaller environments. This has particularly interesting applications to virtual reality.

KinectFusion

One work which has garnered a lot of attention in the reconstruction and modeling field recently is KinectFusion (Izadi et al. 2011). KinectFusion is a framework that allows a user to create a detailed 3D reconstruction of an object or a small environment in real-time using Microsoft Kinect sensor. Applications of this include scanning, augmented reality, and natural user interaction.

Figure 14: KinectFusion. A) user holding a Kinect. B) reconstructed 3D model with camera pose. C) texture-mapped model with simulated particles. D) touch interaction example. E) object segmentation (Izadi et al. 2011).

KinectFusion stores the data in a voxel grid allocated on GPU memory, where each voxel stores the running average of its distance to an assumed object. The algorithm relies heavily on a GPU implementation of its processing pipeline to enable real-time rates. First, the depth map from the sensor is converted into a point cloud. Then, a rigid 6DOF transformation from the sensor's current coordinate frame to the global coordinate frame is computed, using a GPU implementation of the Iterative Closest Point (ICP) algorithm. Having done this, each point is converted into global coordinates, and the voxel grid is updated. Finally, the algorithm does raycasting in order to render the scene to the user.

Since KinectFusion needs to allocate the entire voxel grid on GPU memory, the size of the environment that can be mapped is restricted by available GPU memory as well as the desired resolution. For this reason, KinectFusion is not suitable for SLAM as is. However, there is certainly room to take some of its ideas (GPU implementation of ICP) and adapt others (voxel-grid representation) and to SLAM.

4. Natural User Interfaces (NUIs)

Outside of gaming, the area where cheap RGB-D sensors have generated the most interest and applications is Natural User Interfaces. Here, RGB-D sensors are being taken up both for consumer entertainment applications, as well as in the workplace. For example, Sunnybrook Hospital in Ontario, Canada, has installed the Kinect in operating rooms, allowing surgeons to navigate through medical

images during procedures without having to rescrub (see http://www.youtube.com/watch?v=f5Ep3oqicVU).

5. Libraries and SDKs

This section discusses various libraries and SDKs that have facilities to help developers and researchers use consumer RGB-D sensors.

5.1. Point Cloud Library (PCL)

The Point Cloud Library (PCL) is available at http://www.pointclouds.org/. PCL is an open source library for generic point cloud processing tasks, and includes a simple interface to grab input from the Microsoft Kinect, Asus Xtion, and PrimeSense Reference Design RGB-D sensors. Figure 15 shows an overview of the different modules available in PCL.

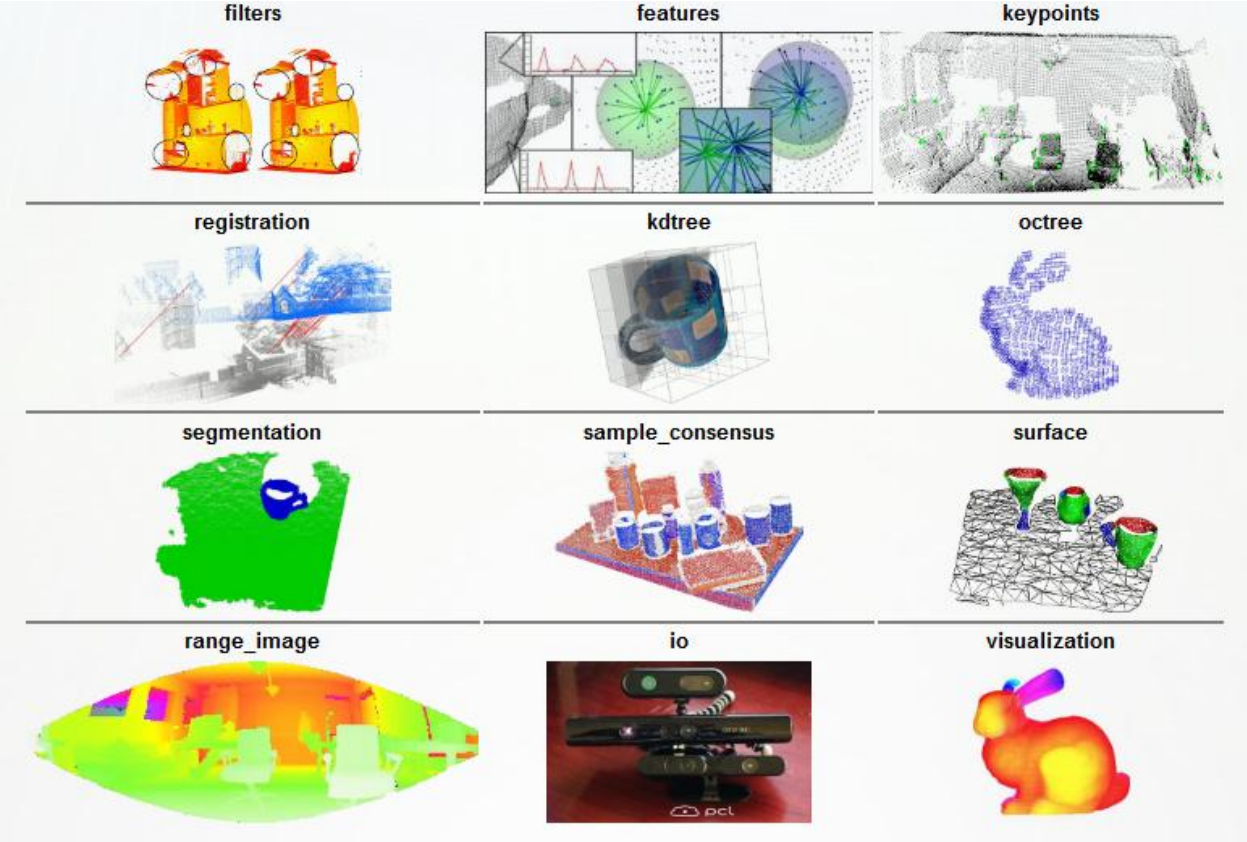

Figure 15: An overview of PCL modules. Source: **http://www.pointclouds.org/documentation/**.

PCL contains tools that are helpful through the entire processing pipeline for RGB-D SLAM, including an input interface, filtering, sample consensus, surface reconstruction, feature extraction, registration, and visualization. Since PCL is still a relatively young project, the documentation is currently an ad hoc collection of tutorials and class interfaces.

PCL is a maintained by Willow Garage, and is free for research and commercial use under the BSD license. The original publication is by Rusu and Cousins (Rusu & Cousins 2011).

5.2. Microsoft Kinect SDK

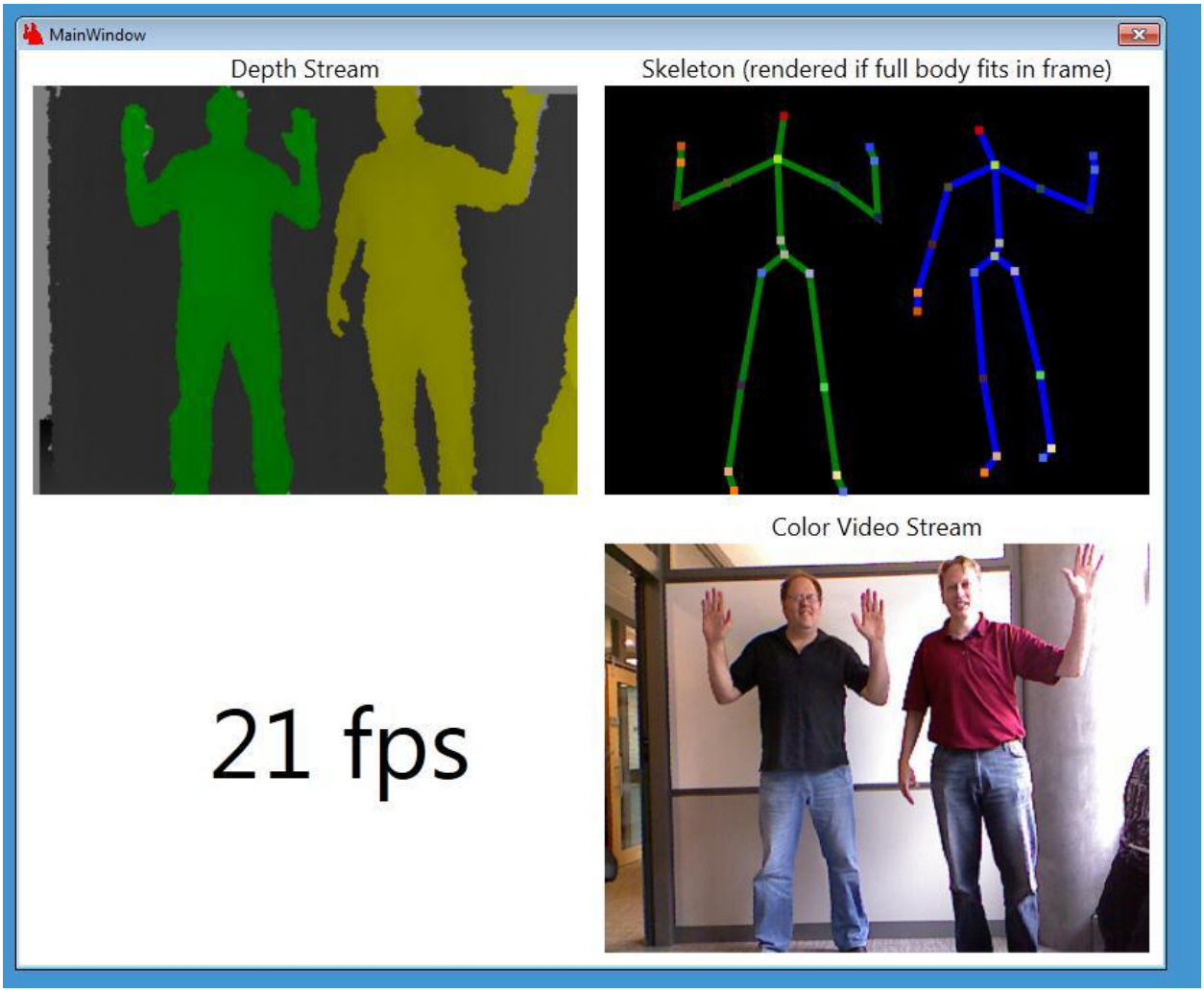

Figure 16: A screenshot of the SkeletalViewer application included in the Microsoft Kinect SDK (source: SkeletalViewer Walkthrough, Microsft Kinect SDK).

Microsoft released the Kinect for Windows SDK (http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx) as a free download. The SDK provides straightforward

access to the Kinect feeds: depth, disparity, and color. In addition, the SDK can also automatically identify up to two people ("players") in the depth stream, and provide a player ID for each point in the depth map. Finally, the SDK also provides access to the microphone array, including speech recognition capabilities in English, French, Spanish, Italian, and Japanese. Clearly, the SDK is geared towards NUI applications.

The SDK is integrated with Microsoft's Visual Studio IDE and comes in C++ and C#; in my experience, the C# version is easier to use.

5.3. OpenCV

The popular OpenCV (Open Source Computer Vision Library, http://opencv.willowgarage.com/wiki/) library provides an interface to access the following streams provided by the Microsoft Kinect: raw depth, RGB depth, and RGB output. Using OpenCV for processing Kinect data has the advantage of being able to use the vast plethora of algorithms implemented in OpenCV. On the other hand, OpenCV deals with the data in only two dimensions, which is inherently limiting for processing 3D data. For more information on using the Microsoft Kinect with OpenCV, visit http://opencv.willowgarage.com/wiki/Kinect.

5.4. OpenNI

OpenNI is an open source library for programming RGB-D devices for NUI applications. It is available from http://www.openni.org/.

References

Besl, P.J. & McKay, H.D., 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), pp.239-256. Available at: http://dl.acm.org/citation.cfm?id=132013.132022 [Accessed March 1, 2012].

Calonder, M. et al., 2010. BRIEF : Binary Robust Independent Elementary Features. In *ECCV*. pp. 778-792.

Du, H. et al., 2011. Interactive 3D modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*. New York, New York, USA: ACM Press, p. 75. Available at: http://dl.acm.org/citation.cfm?id=2030112.2030123 [Accessed March 29, 2012].

Fioraio, N. & Konolige, K., 2011. Realtime visual and point cloud slam. *Proc. of the RGB-D Workshop, Robotics Science and Systems (RSS)*, pp.1-3. Available at: http://www.cs.washington.edu/ai/Mobile_Robotics/rgbd-workshop-2011/camera_ready/fioraio-rgbd11-realtime-slam.pdf [Accessed May 31, 2012].

Henry, P. et al., 2010. RGB-D Mapping: Using depth cameras for dense 3D modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*. Available at: http://ils.intel-research.net/uploads/papers/3d-mapping-iser-10-final.pdf [Accessed September 14, 2011].

Henry, P. et al., 2012. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, p.0278364911434148-. Available at: http://ijr.sagepub.com/cgi/content/abstract/0278364911434148v1 [Accessed March 9, 2012].

Huang, A., Bachrach, A. & Henry, P., 2011. Visual odometry and mapping for autonomous flight using an rgb-d camera. *Int. Symposium on Robotics Research*, pp.1-16. Available at: http://ai.cs.washington.edu/www/media/papers/Huang-ISRR-2011.pdf [Accessed May 31, 2012].

Izadi, S. et al., 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*. Available at: http://dl.acm.org/citation.cfm?id=2047270 [Accessed May 22, 2012].

Khoshelham, K. & Elberink, S.O., 2012. Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, 12(2), pp.1437-1454. Available at: http://www.mdpi.com/1424-8220/12/2/1437/ [Accessed March 1, 2012].

Lange, S. et al., 2012. Autonomous corridor flight of a UAV using a low-cost and light-weight rgb-d camera. *Advances in Autonomous Mini Robots*, pp.183-192. Available at: http://www.springerlink.com/index/HG33P8878036387J.pdf [Accessed June 6, 2012].

Lowe, D.G., 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, pp. 1150-1157 vol.2. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=790410 [Accessed March 10, 2012].

Pfister, H. et al., 2000. Surfels: Surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., pp. 335–342. Available at: http://portal.acm.org/citation.cfm?id=344936 [Accessed October 30, 2011].

Rosten, E. & Drummond, T., 2006. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*. pp. 1-14. Available at: http://www.springerlink.com/index/y11g42n05q626127.pdf [Accessed June 1, 2012].

Rusu, R.B. & Cousins, S., 2011. 3D is here: Point Cloud Library (PCL). In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China*. Available at: http://www.pointclouds.org/assets/pdf/pcl_icra2011.pdf [Accessed September 19, 2011].

Sturm, J. et al., 2011. Towards a benchmark for RGB-D SLAM evaluation. In *Proc. of the RGB-D Workshop in conjunction with RSS*. Available at: http://stephane.magnenat.net/publications/Towards a benchmark for RGB-D SLAM evaluation - Sturm et al. - RSS workshop - 2011.pdf [Accessed June 7, 2012].