

FEASIBILITY STUDY OF USING THE ROBOEARTH CLOUD ENGINE FOR RAPID MAPPING AND TRACKING WITH SMALL UNMANNED AERIAL SYSTEMS

Julien Li-Chee-Ming, Costas Armenakis

Geomatics Engineering, GeoICT Lab
Department of Earth and Space Science and Engineering
Lassonde School of Engineering, York University
Toronto, Ontario, M3J 1P3
{julienli}, {armenc} @yorku.ca

KEYWORDS: sUAS, Building, Close range, Georeferencing, Internet/web, Point cloud, Reconstruction, Registration

ABSTRACT:

This paper presents the ongoing development of a small unmanned aerial mapping system (sUAMS) that in the future will track its trajectory and perform 3D mapping in near-real time. As both mapping and tracking algorithms require powerful computational capabilities and large data storage facilities, we propose to use the RoboEarth Cloud Engine (RCE) to offload heavy computation and store data to secure computing environments in the cloud. While the RCE's capabilities have been demonstrated with terrestrial robots in indoor environments, this paper explores the feasibility of using the RCE in mapping and tracking applications in outdoor environments by small UAMS.

The experiments presented in this work assess the data processing strategies and evaluate the attainable tracking and mapping accuracies using the data obtained by the sUAMS. Testing was performed with an Aeryon Scout quadcopter. It flew over York University, up to approximately 40 metres above the ground. The quadcopter was equipped with a single-frequency GPS receiver providing positioning to about 3 meter accuracies, an AHRS (Attitude and Heading Reference System) estimating the attitude to about 3 degrees, and an FPV (First Person Viewing) camera. Video images captured from the onboard camera were processed using VisualSFM and SURE, which are being reformed as an Application-as-a-Service via the RCE. The 3D virtual building model of York University was used as a known environment to georeference the point cloud generated from the sUAMS' sensor data. The estimated position and orientation parameters of the video camera show increases in accuracy when compared to the sUAMS' autopilot solution, derived from the onboard GPS and AHRS. The paper presents the proposed approach and the results, along with their accuracies.

1. INTRODUCTION

This paper presents the development of a small unmanned aerial mapping system (sUAMS) aiming at self-localization and 3D mapping in near-real time. The main goal is to deliver autonomy related to situational awareness and spatial intelligence in reconnaissance tasks while reducing the workload of the ground operator.

The mapping system is integrated with an Aeryon Scout quadcopter. It is equipped with a GPS sensor that provides positioning to about 3 meter accuracies, and an AHRS (Attitude and Heading Reference System) that estimates attitude to about 3 degrees. The quadcopter is also equipped with an FPV (First Person Viewing) camera, which streams video to a ground control station, giving the operator a perspective view from the aerial vehicle's "cockpit". It is used as a visual aid in piloting the small unmanned aerial vehicle (UAV).

As both mapping and tracking algorithms require powerful computational capabilities and large data storage facilities, an Application-as-a-Service is being developed on top of the RoboEarth Cloud Engine (RCE) to offload heavy computation, store data to secure computing environments in the Internet cloud, and share and re-use data (RoboEarth, 2014). The RoboEarth library provides software components commonly used in robotic applications, such as object databases, object recognition and learning models, and a visual SLAM system that is based on a distributed framework.

RoboEarth has demonstrated its capabilities with terrestrial robots in indoor environments. For example, an omni-wheel service robot was tasked to serve a drink to a patient in a hospital room. The robot first queried the RoboEarth database

for relevant information and downloaded the knowledge previously collected by other robots; such knowledge included object descriptions and instructions on how to complete tasks. The robot then successfully constructed a model of the environment and localized itself, then recognized objects that were downloaded and performed appropriate actions to complete the task (Waibel et al., 2011).

This paper demonstrates that RoboEarth can also be applied to mapping and tracking applications in outdoor environments by sUAMS. At the current stage of development, the presented experiments assess the data processing strategies and evaluate the attainable tracking and mapping accuracies using data obtained by the sUAMS. An Aeryon Scout quadcopter flew over York University, up to approximately 40 metres above the ground, while its onboard camera focused on buildings, walkways, and trees. The 3D virtual building model of York University's Keele Campus was used as a known environment to georeference the map. The model consists of photorealistic reconstructions of buildings, trees, and the terrain.

Video images captured from the onboard camera were reconstructed using VisualSFM (Wu, 2011) and densely matched using SURE (Rothermel et al., 2012). Both are being reformed as an Application-as-a-Service via the RCE. The correct building models were identified from the generated point cloud and the sUAMS localized itself and mapped the 3D environment in a geodetic coordinate system. The estimated position and orientation parameters of the video camera show increases in accuracy when compared to the sUAMS' autopilot solution, derived from the onboard single frequency GPS receiver and MEMS-IMU. The paper presents the proposed approach and the obtained results, along with their accuracies.

2. THE ROBOEARTH CLOUD ENGINE (RCE)

The RoboEarth Cloud Engine (RCE), also called Rapyuta, is an open source robotics Platform-as-a-Service (PaaS) on top of which robotics developers can design robotics Software-as-a-Service (SaaS) applications.

Its main difference to similar PaaS frameworks, like the Google App Engine, is that it is specifically designed for multi-process and high-bandwidth robotics applications, such as mapping and object recognition. Further, the RCE is able to launch almost all of the current 3000+ ROS packages in its cloud environment (Mohanarajah et al., 2014; ROS, 2014).

Similar systems include the DAVinCI project (Arumugam et al., 2010), which used ROS to implement a cloud-based parallel implementation of Fast-SLAM (Thrun et al., 2005). Unfortunately, the DAVinCI project is not publicly available. Riazuelo et al. (2013) presented a collaborative mapping system where they moved the computationally intensive bundle adjustment process of the Parallel Tracking and Mapping (PTAM) algorithm (Klein and Murray, 2009) to a server. Heroku (Lindenbaum et al., 2007) provides access to program APIs and sockets, however it does not allow the server to send data to the robot, and does not allow robots to share information through networked computing environments. Rosbridge (Crick et al., 2012) is open source, and enables communication between a robot and one ROS environment in the cloud (Gherardi et al., 2014).

3. MAPPING AND TRACKING IN THE CLOUD

The following sections describe the ongoing development of a small unmanned aerial mapping system with the objective to track its trajectory and perform 3D mapping of its environment in near-real time. An overview of the system architecture is provided in Figure 1.

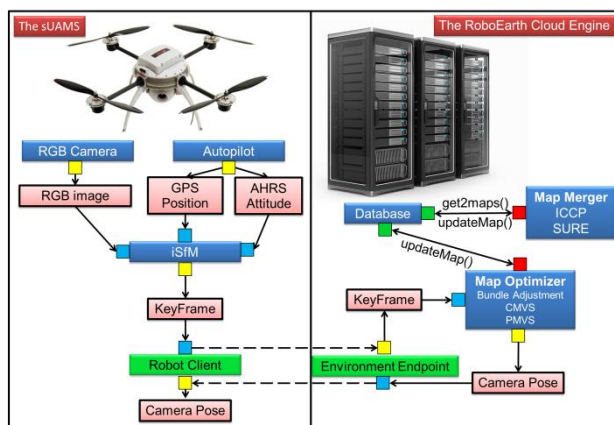


Figure 1: Blue rectangles depict ROS nodes, green rectangles for the RCE's endpoints, red rectangles for topics, small yellow squares for topic publishers, small blue squares for topic subscribers, small green squares for service servers, and small red squares for server clients. The direction of the arrows indicates the flow of data from the publishers to the topics and topics to subscribers, and arrows with labels connect service servers to service clients. Notably, this architecture was derived from the developers of the RCE (Gherardi et al., 2014).

3.1 The small Unmanned Aerial Mapping System (sUAMS)

An embedded board with a multicore ARM processor, running a Linux operating system, is used for computations onboard the sUAMS. The embedded board connects to the RCE through a wireless USB adapter.

The *RGB Camera* ROS node, running on the onboard processor, reads data from the RGB camera and outputs (publishes) an RGB image (topic). Similarly, the *Autopilot* ROS node reads data from sUAMS' sensors (GPS, IMU, magnetometer) and publishes GPS position and AHRS attitude topics. Given the frames produced by *RGB Camera* and the GPS and attitude data from the *Autopilot* node, the *iSfM* ROS node, also running on the onboard processor, estimates the pose of the camera and a 3D point cloud representing the environment. It publishes keyframes to the RCE every time the camera's estimated pose passes a certain threshold relative to the previous keyframe. A keyframe mainly consists of an RGB image, the camera pose, image features and correspondences, and a frame ID.

3.1.1 Incremental Structure from Motion (iSfM)

In the *iSfM* (Wu, 2013) ROS node running on the onboard processor, a two-view reconstruction is first estimated by triangulating successful feature matches between two images. Incoming images are then repeatedly matched and the 3D model is extended from the two-view reconstruction. One image is added at each iteration. In order alleviate error accumulation, partial bundle adjustments (BAs) are run onboard the sUAMS, using a constant number of recently added images (e.g., 20) and their associated 3D points. Following the BA step, filtering removes the points that have large re-projection errors or small triangulation angles. Finally, the next iteration starts or a re-triangulation occurs (Section 3.2.2).

The exterior orientation of the cameras and the 3D point estimations typically converge quickly during reconstruction, thus full BAs (on all cameras and 3D points) are performed in the RCE when the size of a model increases by a certain ratio (e.g. 5%). Although the latter added cameras are optimized by fewer full BAs, there are normally no accuracy problems because full BAs improve more for the least accurate parts. Notably, as the model gets larger, more cameras are added before running a full BA (Wu, 2013).

3.1.2 Feature Matching

Image matching is one of the most time-consuming steps of SfM. Due to the wide range of viewpoints in a large collection of photos, Wu (2013) states that the majority of image pairs do not match (75% - 98%). A large portion of matching time is saved by identifying the good pairs robustly and efficiently. For instance, the approximate GPS tags are used to match images only to the nearby ones (Frahm et al., 2010). Further, pre-emptive feature matching (Wu, 2013) filters correspondence candidates based on the scales of SIFT features (Lowe, 2004), as the chances of correctly matching the top-scale features is higher than matching randomly selected features. Finally, increases in matching speed can be achieved by parallelizing the search with multiple machines (Agarwal et al., 2010) and multiple GPUs (Frahm et al., 2010).

3.2 Inside The RoboEarth Cloud Engine

The *Map Optimizer* ROS node, running in the RCE, (Figure 1) receives the image keyframes as input, and optimizes the

keyframe poses and triangulated 3D point positions. After each full BA, *Map Optimizer* updates the keyframe poses and the 3D point cloud, and then stores them in the *Database*. The *Map Merger* ROS node retrieves two maps from the *Database* and tries to find correspondence that will combine the two maps.

The *Map Optimizer* ROS Node consists of three processes: 1) Bundle Adjustment, 2) Clustering views for multi-view stereo (CMVS), and 3) Patch-based multi-view stereo (PMVS). These processes have been moved to the cloud because they are computationally intensive, but have been optimized through parallel computing. The following sections describe each process.

3.2.1 Bundle Adjustment (BA)

A set of measured image feature locations and correspondences are input into a bundle adjustment, with the goal to find the triangulated 3D point positions and camera parameters that minimize the reprojection error (Triggs et al, 1999). This optimization problem is treated as a non-linear least squares problem, where the error is the squared norm of the difference between the observed feature location and the reprojection of the corresponding 3D point on the image plane of the camera. The Levenberg-Marquardt (LM) algorithm (Nocedal and Wright, 2000) is used for solving non-linear least squares problem.

Wu et al. (2011) demonstrated a CPU-based BA that is up to 10 times and a GPU-based system that is up to 30 times faster than the current state of the art. The RCE takes advantage of this parallelized BA by spawning multiple secure computing environments and connecting them to build parallel computing architectures on the fly.

3.2.2 Re-triangulation

When GPS is not available, the online *iSfM* solution is prone to drift because of the accumulated errors of relative camera poses. The initially estimated poses and even the poses after a partial BA may not be accurate enough and this may result in some correct feature matches failing a quality test. As the drift is attributed mainly to the accumulated loss of correct feature matches, failed feature matches are re-triangulated (Wu, 2013) when the model size increases (e.g., by 25%). After re-triangulating, a full BA and point-filtering is run to improve the reconstruction. This strategy of reducing the drift is analogous to loop-closing.

3.2.3 Clustering Views for Multi-View Stereo (CMVS)

Multi-view Stereo (MVS) algorithms aim to correlate measurements from a collection of images to derive 3D surface information. Many MVS algorithms reconstruct a single 3D model by using all the images available simultaneously. As the number of images grows the processing time and memory requirements become infeasible. To solve this problem, subsets of overlapping images are clustered into manageable pieces that are processed in parallel, and then the resulting reconstructions are merged (Furukawa et al., 2010). The clustering algorithm is designed to satisfy the following three constraints: 1) redundant images are excluded from the clusters, 2) each cluster is small enough for an MVS reconstruction (a size constraint determined by computational resources), and 3) MVS reconstructions from these clusters result in minimal loss of content and detail compared to that obtained by processing the full image set.

Having extracted image clusters, a patch-based MVS software (PMVS) is used to reconstruct 3D points for each cluster independently.

3.2.4 Patch-Based Multi-View Stereo (PMVS)

The patch-based multi-view stereo (PMVS) algorithm (Furukawa and Ponce, 2011) represents scene surfaces by collections of small oriented 3D rectangular patches (essentially local tangent planes). The algorithm consists of a simple match, expand, and filter procedure:

1) *Matching*: Features found by Harris (Harris and Stephens, 1988) and difference-of-Gaussians operators (Lowe, 2004) are first matched within each cluster of pictures, yielding a sparse set of patches associated with salient image regions. A matching patch is considered to be an inlier if the search along the epipolar lines of other images yields low photometric discrepancies (one minus the normalized cross correlation score) in a minimum number of images (e.g. 2 or 3). Given these initial matches, the following two steps are repeated.

2) *Expansion*: The initial matches are spread to nearby pixels and obtain a dense set of patches.

3) *Filtering*: Visibility constraints are used to eliminate incorrect matches lying either in front or behind the observed surface.

3.2.5 Map Merger

The *Map Merger* ROS node retrieves maps from the *Database* and attempts to merge them into a larger map using a modified version of the Iterative Closest Point (ICP) algorithm (Besl and McKay, 1992). Further, the density of the point clouds are increased using SURE (Rothermel et al., 2012), a dense stereo matching software tool based on the semi-global matching (SGM) algorithm (Hirschmüller, 2008).

3.2.5.1 Iterative Closest Point (ICP)

Given two point clouds that are roughly aligned, ICP uses all of the points to refine their relative transformation. Correspondence is established by pairing points in one point cloud with the closest points in the other. The initial alignment is provided by metre-level GPS positioning in this application. Several additional heuristics have been developed to address the problem of not knowing the extent of overlap and avoiding false point matches. The approach used here is similar to Iterative Closest Compatible Point (ICCP) algorithm (Godin et al., 1994), where points are matched only if their associated feature (colour, normal vector, etc.) are within a given threshold. ICCP can be interpreted as a set of rigidly coupled ICP subproblems between subsets of mutually compatible points. ICCP is equivalent to ICP when all points have compatible attributes.

Further, the point-plane (Chen and Medioni, 1992) method is chosen over the original point-point ICP as it converges an order of magnitude faster, it is more robust against outliers, and produces more accurate results (Pulli, 1999).

3.2.5.2 SURE: Photogrammetric surface reconstruction

The SURE algorithm densifies a point cloud by using the oriented images generated from the bundle adjustment. These images are first rectified to generate epipolar images and dense disparities are calculated across stereo pairs using SGM.

Briefly, the SGM algorithm performs the image alignment required to estimate disparities by the maximizing the mutual information (i.e. minimizing the joint entropy) between two overlapping images. Instead of using the entire image in this calculation (global matching), 16 one-dimensional directional paths are constructed to approximate the image (semi-global matching).

3D points or depth images are then triangulated from the stereo models. Finally, redundant depth measurements are used to remove outliers and increase the accuracy of the depth measurements.

4. DATA COLLECTION

Initial testing was carried out by processing video images with VisualSFM and SURE. The video was collected by a Photo3S high resolution camera onboard of an Aeryon Scout quad-copter UAV (Aeryon, 2013). The UAV flew over York University, up to approximately 40 metres above the ground level (AGL), while the gyro-stabilized camera focused on buildings, walkways, and trees. The imagery was downsampled from 12 megapixels to 1 megapixel to enable fast image processing.

The 3D virtual building model of York University's Keele campus (Armenakis and Sohn, 2009) was used as a known 3D map environment (Figure 2). The model consists of photorealistic 3D polygon reconstructions of buildings, trees, and terrain, generated from building footprint vector data, Digital Surface Model (DSM) with 0.75m ground spacing, corresponding orthophotos at 0.15 m spatial resolution and terrestrial images. The 3D building model was further refined with airborne lidar data having a point density of 1.9 points per square metre (Corral-Soto et al., 2012). This 3D CAD model serves two purposes in the proposed approach. Firstly, it provides the necessary level of detail such that individual buildings can be uniquely identified via point cloud matching. Secondly, it provides ground control points to achieve photogrammetrically sub-meter accuracies of the positional elements of the exterior orientation. The geometric accuracy of the building models is in the order of 10 to 40 cm.



Figure 2: York University's 3D virtual campus model

5. 3D BUILDING MODEL TO 3D POINT CLOUD

The RCE's communication server allows robotic platforms to wirelessly access the RoboEarth knowledge repository. This provides services to download, upload, update, delete, and query object models and maps that are stored in the RoboEarth repository. As such, the York University's georeferenced 3D building models were loaded into the RoboEarth database. If a building model is recognized in the sUAMS' image-derived point cloud, it is used either as photogrammetric ground control to georeference the sUAMS' camera trajectory and the 3D point

cloud, or to improve the georeferencing accuracy provided by the onboard GPS receiver.

To adhere to RoboEarth's object modelling paradigm, each building in the York University campus model was converted to a point cloud before being loaded into RoboEarth's object recognition database. To generate each building's point cloud, the 3D polygon faces were converted to triangular surfaces, then each triangle was sampled at a specified point interval (e.g. 50 cm point spacing was used in the presented experiments). Figures 3 and 4 show the Lassonde and the Vari Hall building models, respectively, along with their corresponding point clouds.

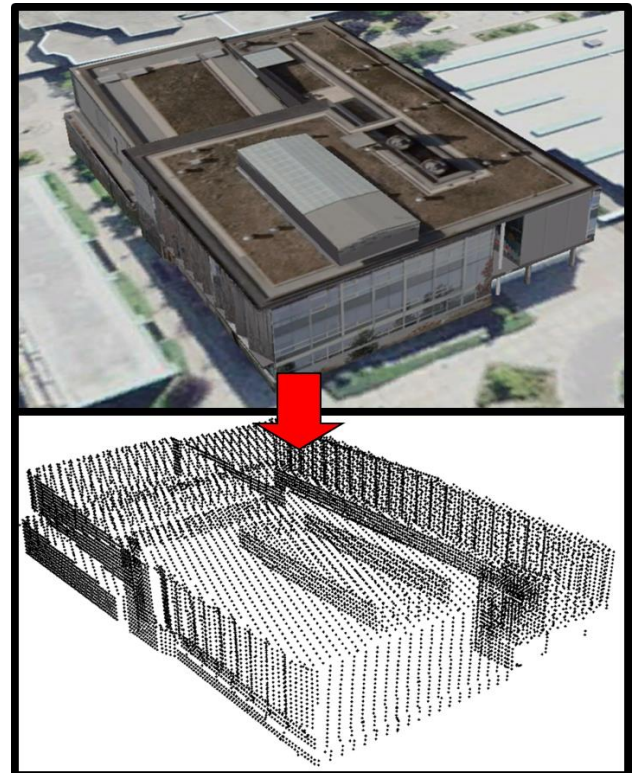


Figure 3: Converting 3D polygon building models to 3D point clouds: The Lassonde Building.

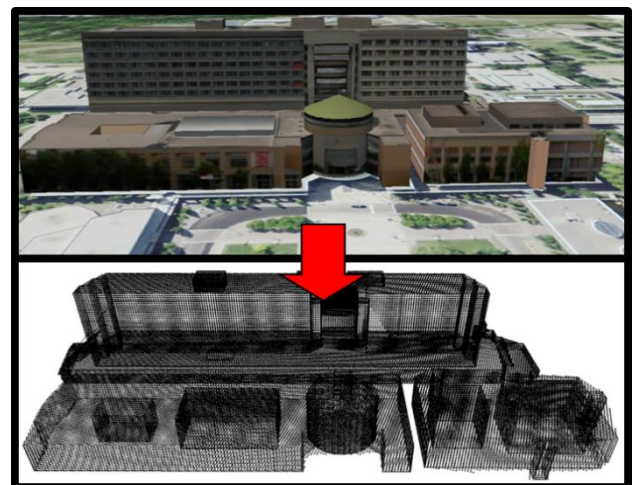


Figure 4: Converting 3D polygon building models to 3D point clouds: The Vari Hall building.

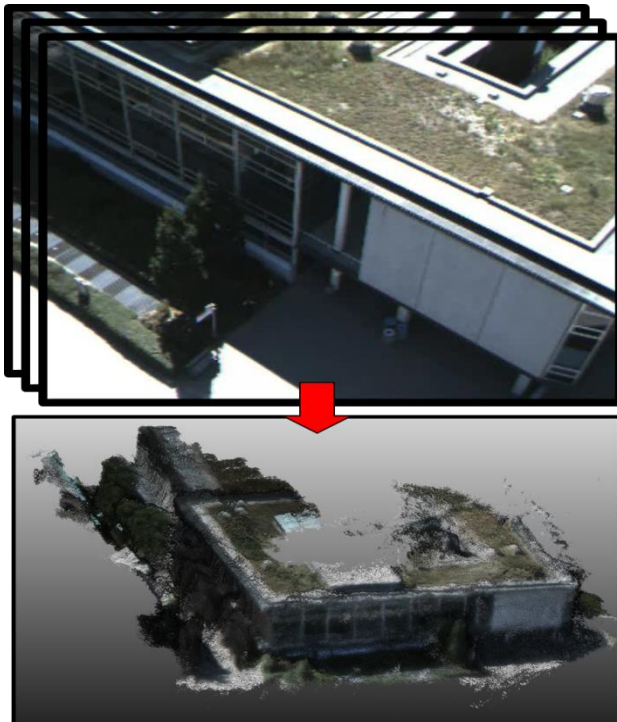


Figure 5: VisualSFM and SURE used to generate a dense point cloud of the Lassonde building from the sUAMS' video imagery.

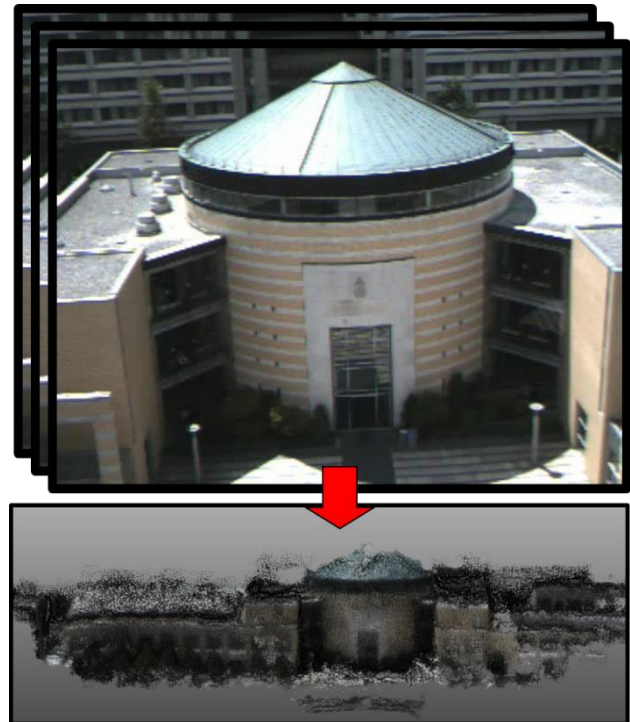


Figure 6: VisualSFM and SURE used to generate a dense point cloud of Vari Hall from the sUAMS' video imagery.

5. VisualSFM and SURE

VisualSFM (Wu, 2011) and SURE (Rothermel et al., 2012) were used to generate dense 3D point clouds from the sUAMS' video imagery (Figures 5 and 6). The reconstruction system integrates all of the above mentioned algorithms. Explicitly, a GPU-based SIFT module parallelized matching, a multicore and GPU-based SfM module estimated the camera parameters and generated a sparse point cloud, and the PMVS/CMVS tool chain efficiently densified the sparse point cloud. SURE was then used to further densify the points clouds.

5. POINT CLOUD GEOREFERENCING

The GPS coordinates stored in the image tags were used to transform the point cloud to a geodetic coordinate system. These coordinates were noisy ($\sigma_{3D} \approx \pm 3m$) as they were provided by the onboard single frequency GPS receiver. To increase the georeferencing accuracy, the sUAMS' point cloud was used to search the database for the corresponding York University building model point cloud. The search matched the geodetic locations of the buildings.

Once the building was identified, the ICCP algorithm detailed in Section 3.2.5.1 was used to improve the geographic position of the sUAMS' point cloud. ICCP iteratively refined the rigid-body transformation between the SfM point cloud and the 3D building model point cloud by repeatedly generating pairs of corresponding points and minimizing the point Euclidean distance error metric.

Firstly the point clouds were segmented into groups of similar normal vectors. Figures 7 (the Lassonde building) and 8 (Vari Hall) show the three groups: points with normal vectors (1) pointing in the positive Y direction ($\pm 45^\circ$), (2) pointing in the negative X direction ($\pm 45^\circ$), and (3) pointing in the positive X direction ($\pm 45^\circ$). For each group, a point-plane ICP was applied, and points with the smallest residuals were used to further refine the estimated transformation parameters. Figure 9 shows the registered point clouds for the Lassonde building (top) and Vari Hall (bottom).

6. ACCURACY ASSESSMENT

Check points were used to assess the accuracy of the georeferenced point clouds generated from the sUAMS' sensor data. The ground truth coordinates were extracted from the York University 3D campus model. The results from manually comparing corresponding points between the sUAMS' point cloud and building model are in Table 1.

Table 1: Accuracy assessment of the generated point clouds

Building	# Check Points	RMSE [m]	Average Error [m]
Lassonde	7	0.32	0.29
Vari Hall	13	0.73	0.69

Registering the sUAMS' point cloud with the York University's 3D building model resulted in sub-meter mapping accuracies (root mean square error (RMSE)), an improvement from the georeferencing provided by the onboard GPS receiver (3 meters in positioning accuracies) and the AHRS (3 degrees in attitude accuracies).

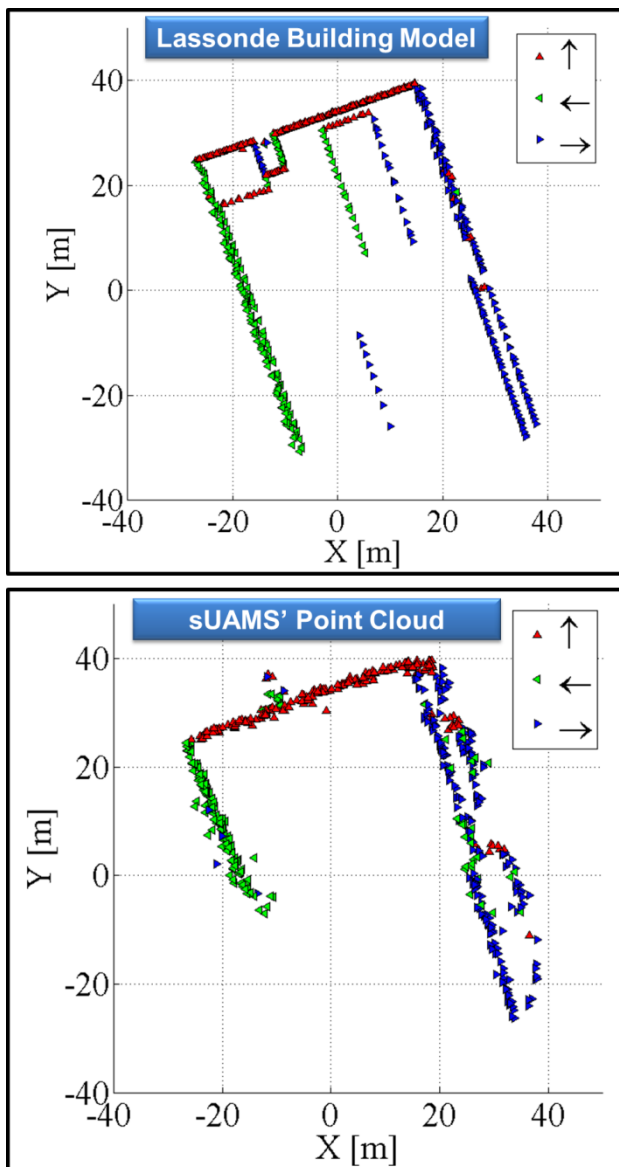


Figure 7: Segmentation of the Lassonde building model (top) and corresponding point cloud generated from the sUAMS' data (bottom). Points are segmented into three groups based on their normal vector: Group 1 (red triangles): normals pointing in the positive Y direction ($\pm 45^\circ$), Group 2 (green triangles): normals pointing in the negative X direction ($\pm 45^\circ$), and Group 3 (blue triangles): normal pointing in the positive X direction ($\pm 45^\circ$).

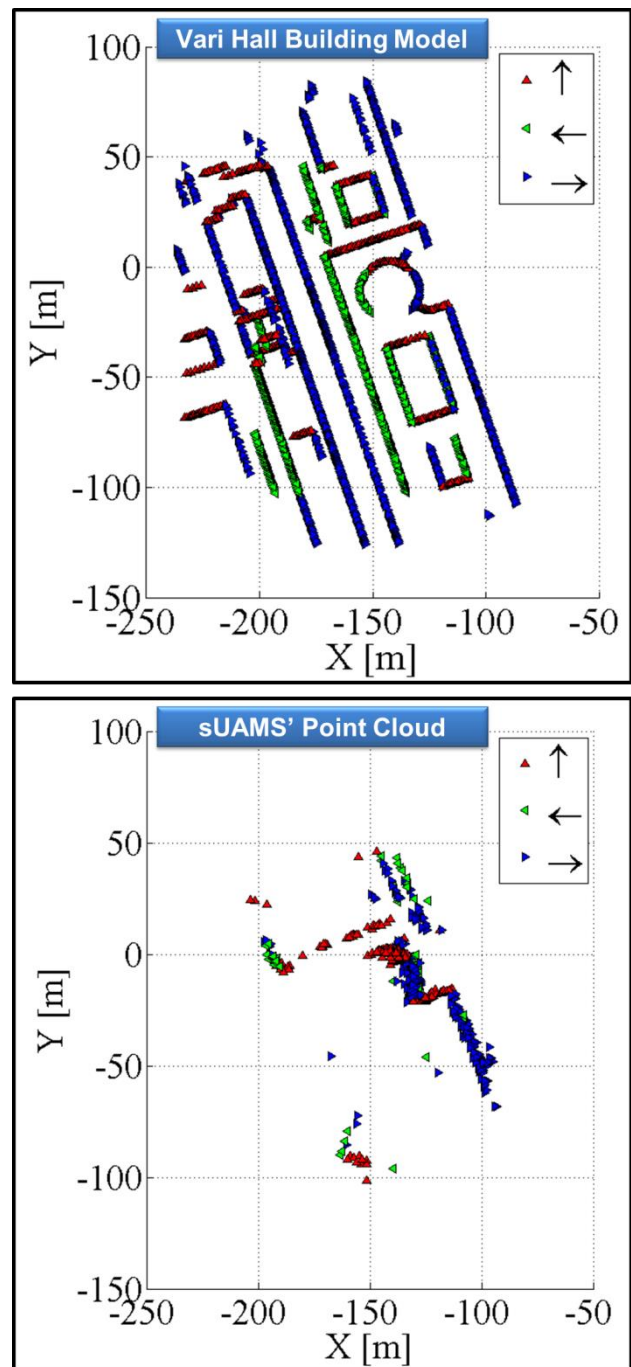


Figure 8: Segmentation of the Vari Hall building model (top) and corresponding point cloud generated from the sUAMS' data (bottom). Points are segmented into three groups based on their normal vector: Group 1 (red triangles): normals pointing in the positive Y direction ($\pm 45^\circ$), Group 2 (green triangles): normals pointing in the negative X direction ($\pm 45^\circ$), and Group 3 (blue triangles): normal pointing in the positive X direction ($\pm 45^\circ$).

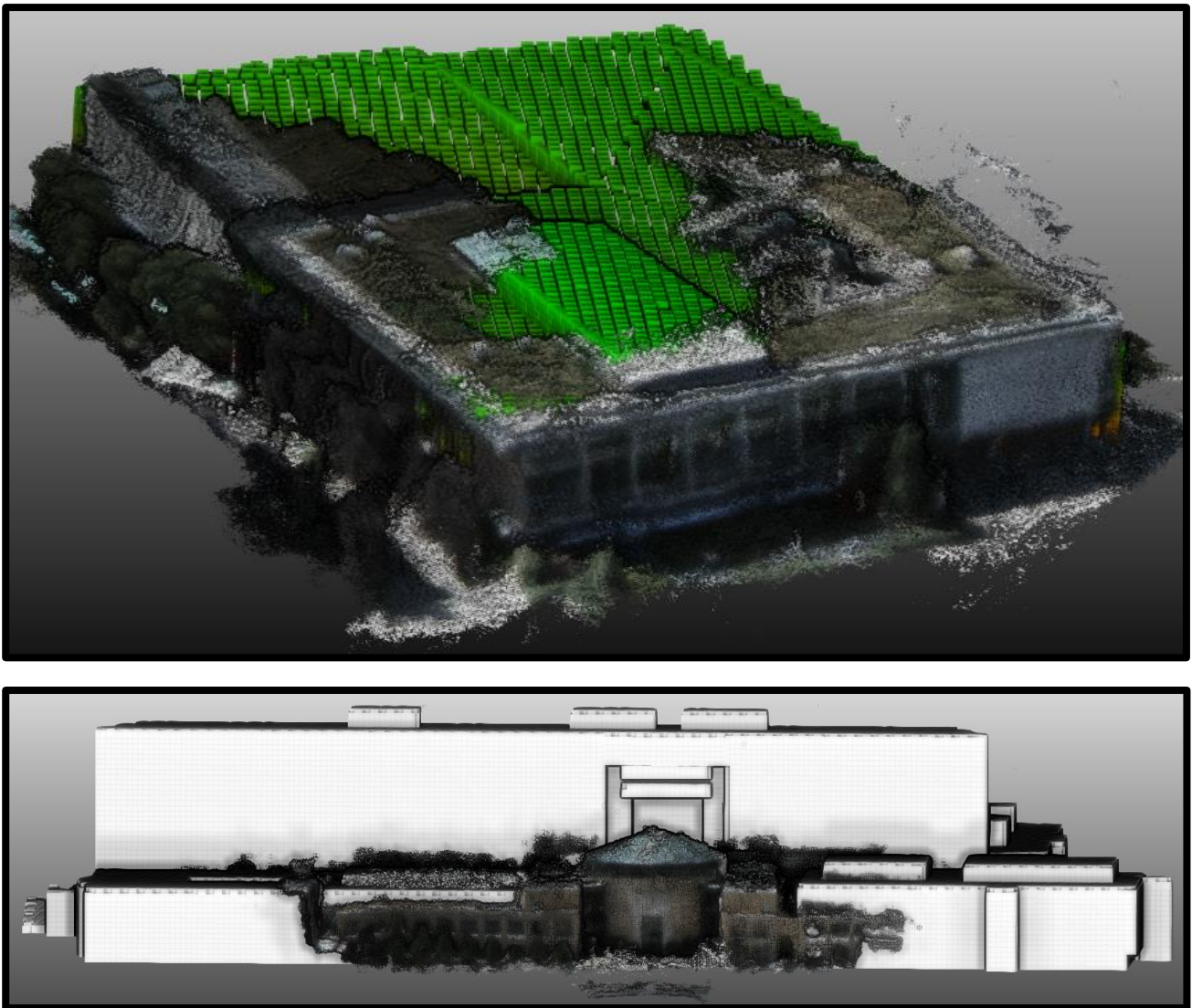


Figure 9: Registered point cloud after ICCP. Top: The Lassonde building. Bottom: The Vari Hall building.

7. CONCLUSTIONS & FUTURE WORK

This paper presents the ongoing development of a small unmanned aerial mapping system (sUAMS) that, in near-real time, will track its trajectory and perform 3D mapping. An approach was developed to offload the computationally expensive mapping and tracking processes to the RoboEarth Cloud Engine (RCE). While RoboEarth has demonstrated the RCE's capabilities with terrestrial robots in indoor environments, this project is exploring the feasibility of using the RCE in mapping and tracking applications in outdoor environments by small UAVs.

Initial testing used VisualSFM and SURE to generate point clouds of buildings from imagery collected from an Aeryon Scout UAV. The correct building models were recognized from the point cloud and the UAV localized itself and mapped the 3D environment. The sUAMS' derived building point clouds resulted in sub-meter mapping accuracies, indicating that the estimated position and orientation parameters of the video camera are of improved accuracy when compared to the UAV's autopilot solution, derived from the onboard GPS and AHRS.

Future work involves moving forward from the testing phase and implementing the real-time unmanned aerial mapping system (sUAMS). Finally, more robust object recognition techniques will be adapted to more accurately and reliably match the model and SfM point clouds.

ACKNOWLEDGEMENTS

NSERC's financial support for this research work through a Discovery Grant is much appreciated. Special thanks are given to the members of York University's GeoICT Lab Damir Gumerov, Yoonseok Jwa, Brian Kim, and Yu Gao, who contributed to the generation of the York University 3D Virtual Campus Model. We also thank Professor James Elder's Human & Computer Vision Lab in York University's Centre for Vision Research for providing the UAV video data.

REFERENCES

- Aeryon. 2013. Aeryon Scout Brochure. www.aeryon.com/products/avs/aeryon-scout.html. (accessed: 14-Sep-2014).
- Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., and Szeliski, R. 2009. Building Rome in a day. In ICCV.
- Armenakis, C., and Sohn, G. 2009. iCampus: 3D modeling of York University campus. In Proc. 2009 Conf. of the American Society for Photogrammetry and Remote Sensing, Baltimore, MA, USA.
- Arumugam, R., Enti, V.R., Baskaran, K., and Kumar, A.S. 2010. DAVinCi: A cloud computing framework for service robots. In Proc. IEEE Int. Conf. Robotics and Automation. IEEE, pp. 3084–3089.
- Besl, P. and McKay, N. 1992. A method for registration of 3-D Shapes. Trans. PAMI, 14(2).
- Chen, Y., and Medioni, G. 1992. Object modelling by registration of multiple range images. Image and Vision Computing, 10(3): 145-155.
- Corral-Soto, E.R., Tal, R., Wang, L., Persad, R., Chao, L., Chan, S., Hou, B., Sohn, G., and Elder, J.H. 2012. 3D Town: The automatic urban awareness project. In Proc. 9th Conf. on Computer and Robot Vision, Toronto, pp. 433-440.
- Crick, C., Jay, G., Osentoski, S., and Jenkins, O.C. 2012. ROS and rosbridge: Robotists out of the loop. In Proc. Annual ACM/IEEE Int. Conf. Human-Robot Interaction, pp. 493–494.
- Frahm, J., Fite Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, J., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. 2010. Building Rome on a cloudless day. In ECCV, pages IV: 368–381.
- Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R. 2010. Towards internet-scale multi-view stereo. In CVPR.
- Furukawa, Y., and Ponce, J. 2011. Accurate, dense, and robust multi-view stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence.
- Gherardi, L., Hunziker, D., and Mohanarajah, G. 2014. A software product line approach for configuring cloud robotics application. In Proc. IEEE Cloud 2014.
- Godin, G., Rioux, M., and Baribeau, R. 1994. Three-dimensional registration using range and intensity information. Proc. SPIE Videometrics III, vol. 2350, pp. 279-290.
- Harris, C., and Stephens, M.J. 1988. A combined corner and edge detector. In Alvey Vision Conference, pp 147–152.
- Hirschmüller, H. 2008. Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence 30, pp. 328-341.
- Klein G., and Murray, D. 2009. Parallel tracking and mapping on a camera phone. In Proc. IEEE Int. Symp. Mixed and Augmented Reality, pp. 83–86.
- Lindenbaum, J., Wiggins, A., and O. Henry, O. 2007. Heroku. www.heroku.com. (accessed: 14-Sep-2014).
- Lowe, D. G. 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision.
- Mohanarajah, G., Hunziker, D., Waibel, M., and D'Andrea, R. 2014. Rapyuta: A cloud robotics platform. IEEE Transactions on Automation Science and Engineering.
- Nocedal, J., and Wright, S. 2000. Numerical optimization. Springer.
- Riauelo, L., Civera, J., and Montiel, J.M.M. 2013. C2tam: A cloud framework for cooperative tracking and mapping. Robotics and Autonomous Systems.
- RoboEarth. 2014. A World Wide Web for Robots, www.roboearth.org. (accessed: 14-Sep-2014).
- ROS. 2014. Robot Operating System. www.ros.org. (accessed: 14-Sep-2014).
- Rothermel, M., Wenzel, K., Fritsch, D., Haala, N. 2012. SURE: Photogrammetric surface reconstruction from imagery. Proceedings LC3D Workshop, Berlin.
- Pulli, K. 1999. Multiview registration for large data sets. Proc. 2nd Int'l Conf. 3-D Digital Imaging and Modeling, Ottawa, pp. 160-168.
- Thrun, S., Burgard, W., and Fox, D. 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.
- Triggs, B., McLauchlan, P., R.I. H., and Fitzgibbon, A. 1999. Bundle Adjustment - A modern synthesis. In Vision Algorithms'99, pp. 298–372.
- Waibel, M., Beetz, M., Civera, J., d'Andrea, R., Elfving, J., Galvez-Lopez, D., Häussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle O., and Van de Molengraft, M.J.G. 2011. RoboEarth – A World Wide Web for Robots. In Robotics & Automation Magazine, IEEE, 18(2): 69-82.
- Wu, C. 2011. VisualSFM: A Visual Structure from Motion System. www.ccwu.me/vsfm. (accessed: 14-Sep-2014).
- Wu, C. 2013. Towards linear-time incremental structure from motion. In 3DV, 2013.
- Wu, C., Agarwal, S., Curless, B., and Seitz, S.M. 2011. Multicore bundle adjustment. In CVPR.