STAG: Smart Tools & Apps for Graphics (2015)
Silvia Biasotti and Marco Tarini and Andrea Giachetti (Editors)

Low-cost Experimental Setups for Mid-air 3D Reconstruction

Alexandru Dancu[1], Marco Fratarcangeli[1], Mickaël Fourgeaud[1], Zlatko Franjcic[2,1], Daniel Chindea[3], Morten Fjeld[1]

[1]Chalmers University of Technology, Sweden [2]Qualisys AB, Sweden [3]highimage.se

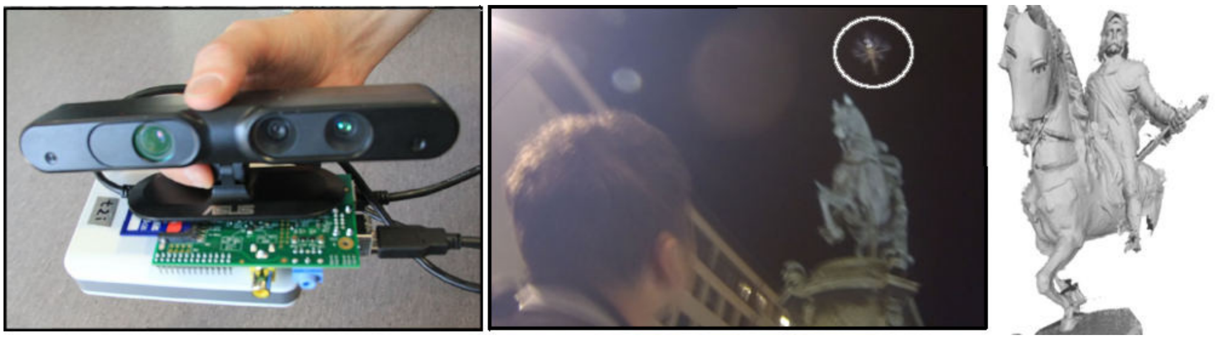

Figure 1: Reconstruction of a statue using a multicopter: (*left*) handheld ASUS Xtion depth sensor - Raspberry Pi - battery system; (*middle*) remotely controlled multicopter while recording depth maps; (*right*) reconstructed 3D model of the statue.

Abstract
The reconstruction of the physical environment using a depth sensor involves data-intensive computations which are difficult to implement on mobile systems (e.g., tracking and aligning the position of the sensor with the depth maps). In this paper, we present two practical experimental setups for scanning and reconstructing real objects employing low-price, off-the-shelf embedded components and open-source libraries. As a test case, we scan and reconstruct a 23 m high statue using an octocopter without employing external hardware.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—; I.4.5 [Image Processing and Computer Vision]: Reconstruction—

1. Introduction

Multicopters are increasingly used in novel outdoor applications, such as package delivery (Swiss Post [drob], Amazon Prime Air [droa]), displaying information in mid-air [SAS*14], and around mobile users [Luk14]. However, as highlighted in DARPA's Urban Challenge [BIS09], GPS information alone is not sufficient to navigate accurately both unmanned aerial vehicles (UAV) such as multicopters, and unmanned ground vehicles (UGV). Precision localisation in urban environments is made possible only by correlating range images with GPS, inertial measurement units (IMUs), and wheel odometry [LMT07].

Existing approaches for sensing the surrounding environment using multicopters and aerial devices can rely on RGB cameras [ESC14, BBBE14], complex fusion algorithms employing a combination of laser range-finders, stereo and monocular color cameras [BPHR11,BBR12], and laser scanning technologies [CCD*11]. In some cases, however, depth sensors are preferred over normal RGB cameras because they do not suffer from poor lighting conditions, it is easier to segment different parts of the scene (e.g., separate the foreground from the background), and thus they support robust algorithms in scene understanding.

Equipping multicopters with depth sensors can effectively support registration of environment geometries, enabling improved navigation capabilities and new applications and services. For example, there are already ongoing initiatives aiming to create connected and sustainable cities and some methods are to utilize networks of fixed and mobile sensors [icr]. Data from urban sensors and mobile phones can be aggregated and could be used in urban analysis studying urban dynamics and offering location-based services [RWFP06]. The required data of the urban state can be collected in mainly two ways: installing fixed sensors and aggregating data from moving sensors roaming in the environment. Using multicopters clearly offers an additional method to collect the required environmental data.

As we synthesise the engineering challenges raised in the area of navigation and in the area of cartographic services, we suggest that both these, and further areas (e.g., cultural heritage [MFE13, PCGS15]), present the following research questions (RQ) worthy of scientific investigation:

RQ1: What are the criteria for successful low-cost strategy for acquiring 3D geometry from a multicopter hardware setup enabling real-time reconstruction?

RQ2: What is an efficient strategy for online analysis of a reconstructed mesh enabling mesh correction and rescanning if needed?

In this paper, we describe two hardware setups based on registration tools using components that are low-cost and widely available on the market. In the first experiment, we use a Raspberry Pi connected to an ASUS Xtion depth sensor. Depth maps are recorded and then downloaded and reconstructed offline. The obtained meshes are visualized and further aligned manually. In the second experiment, we test the real-time reconstruction capabilities of the mobile parallel platform Jetson K1.

2. Related work

We review applications of real-time reconstruction performed on the ground indoors or outdoors, and research employing multicopters where SLAM is used in navigation and environment mapping.

Low-cost depth sensors support new types of applications based on real-time reconstruction of the physical environment. Google Tango [tan] attempts to give mobile devices an understanding of space and motion by integrating depth sensors and combining it with accelerometer and gyroscope data of mobile phones. Chow et. al [CLH*14] propose a mobile mapping system for indoor environments that integrate IMU, depth maps from two Kinects, and range images from a LiDAR system.

The wide availability of multicopter systems support the acquisition of images from hovering or flying objects and pose new research challenges. Several autonomous quadrotor platforms have been developed that are able to navigate, map, and explore the environment using stereo vision and run SLAM off-board [FHH*12, HHL*14].

Indoor real-time reconstruction was performed by streaming depth maps through a wifi connection from an ASUS Xtion sensor mounted on a quadcopter [Hub14]. More recently, a multicopter platform has been developed that runs SLAM real-time and can integrate the data from a team of microaerial vehicles into a global mapping and localization at 1 Hz [SAD*14].

Another approach is to use monocular RGB vision and create inexpensive 3D models of buildings through aerial photography using a GoPro camera mounted on multicopter [BBBE14]. This approach is similar to a commercial free service called ReCap 360 [rec] that uses cloud-computing to create 3D models of environments from photographs taken with a GoPro camera and a multicopter.

In contrast, we present two setups not relying on any WIFI connection, using a depth sensor instead of RGB camera, and thus more robust to lighting conditions. Furthermore, our second setup (Sec. 3.2) is able to reconstruct a scanned object online without using external computers.

3. Experiments

We present two setups for mid-air reconstruction to obtain range images from depth sensors using an octocopter. Both the setups are based on off-the-shelf hardware components and open-source software. As a test case, we reconstructed a sculpture which is 23.3 meters high and, as such, it is impractical to scan without an aerial device (Fig. 2). In the first experiment (Sec. 3.1), the depth maps are recorded during the flight, and then the reconstruction press is performed *offline* on a laptop computer with CUDA capabilities. The second experiment (Sec. 3.2) is designed to perform *real-time* reconstruction directly during the flight. In both the experiments, we used the Kinect Fusion [NDI*11, IKH*11] reconstruction method implemented in the Point Cloud Library (PCL) [AMT*12].

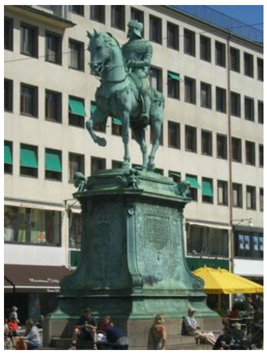
Figure 2: The scanned statue.

3.1. Offline reconstruction with Raspberry Pi

In this first experiment, we recorded range images with a depth sensor connected to a Raspberry Pi (a small, low-cost computer) [UH14], mounted on the octocopter. There are

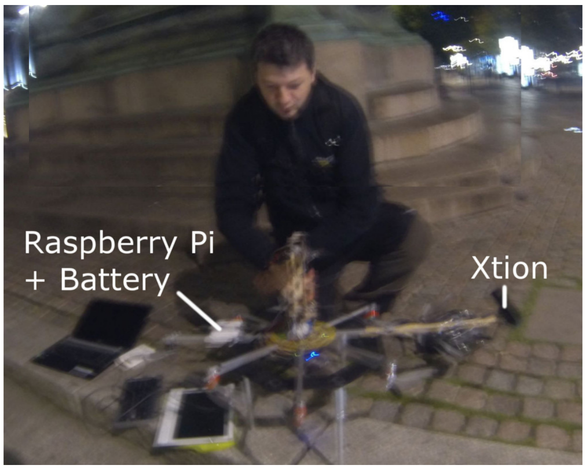

Figure 3: Components of the mid-air reconstruction system.

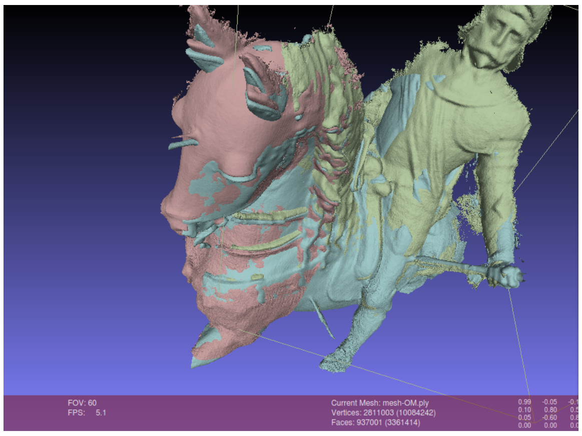

Figure 4: A screenshot of Meshlab [CCC*08] during the manual mesh alignment.

mainly two low-cost depth sensors available on the market, namely the Microsoft Kinect [Zha12] and the ASUS Xtion [GJRVF*13]. Since the Kinect has a significantly bigger weight (810g) and power consumption (3.4W) compared to Xtion (220g and 2.5W), we employed the Xtion sensor.

3.1.1. System components

Our system uses low-price components: the Raspberry Pi (RPi) [UH14] and the ASUS Xtion sensor [GJRVF*13] to acquire and record depth maps. RPi is a $35 credit-card sized computer equipped with a 700 MHz CPU and 500 MB RAM. The Raspbian operating system (OS) is basically a distribution of Linux optimized for the RPi. The open-source library OpenNI [Fal13] is used to acquire the depth maps.

We connected the Xtion to the RPi through one of the two USB ports. However, problems with SD cards threatened the stability of the system. These issues were investigated and tests were made in order to determine the state of the system. Because of the great number read-write cycles required by the compilation process, several cards got corrupted. A solution was found by connecting a flash drive and compiling and recording depth maps on it instead of the SD card from where the OS was running.

The octocopter frame was built from aluminum rods mounted in a star-shape. At the ends of the eight rods the brushless motors Turnigy 2216 were mounted. The flight controller was Mikrokopter [mks]. The Lithium Polymer 4s battery of 14.8V and 5A supported a 10 minute flight time. Two batteries were used. One rod was extended and the Xtion was mounted on it. For balance, on the opposite rod a 5V battery connected to the RPi from Fig. 3 were mounted.

3.1.2. Procedure and stages of reconstruction

The steps required for this experiment were the following:

Record depth maps: The depth maps were recorded on the flash drive using the OpenNI library [Fal13] as an ONI file. The front of the statue was of interest with the the body of the horse rider and the horse front. The octocopter was flown remotely by looking up at it (Fig. 1, middle). After each flight, the octocopter landed and the recording process restarted.

Reconstruct with Kinect Fusion: The ONI files were then loaded in Kinect Fusion and the mesh patches were acquired examining various sequences of the video. Since the octocopter moved continuously and the scanned volume was large, the amount of acquired did not fit in the available memory. Hence, the reconstruction process had to be often restarted, and the resulting 8 meshes were obtained from sequences of a couple seconds.

Manual mesh alignment: The meshes obtained in the previous stage have been aligned and visualized in Meshlab [CCC*08] (Fig. 4). This manual process took several hours and was mainly due to the loss in camera tracking and the limited reconstruction volume of Kinect Fusion.

3.1.3. Speedup of recording depth maps

We customized the RPi hardware configuration, so that the dynamic cpu frequency driver is disabled (`force_turbo=1`), and the frequency of the CPU, GPU processor core and RAM are overclocked. The Raspberry Pi system configuration parameters, that are usually in the BIOS settings, are stored in the folder `/boot/config.txt` file on the SD card. Table 1 summarizes some filesystem, memory, and processor configurations. These are settings which were tested and incrementally adjusted. The following options raised the ONI file recording performance from 4 to 7 fps:

- `force_turbo=1`

- `over_voltage=0`
- `core_freq=350`
- `arm_freq=800`
- `sdram_freq=600`

In particular, increasing the CPU and SDRAM frequency increased the performance to 10 – 15 fps. To compensate the consequent over-heating, we mounted an additional heatsink. Finally, we found that recording speed increases over 20 fps when an externally mounted USB memory is emplyed instead of the SD card where the OS resides.

3.1.4. Lessons learned

Using the described set-up, the actual 3D reconstruction takes place offline and there is not any real-time visual feedback of the scanning process. Hence, the user is not able to assess the quality of the 3D model while the octocopter is flying to 10–15m from the ground, e.g., the presence of holes in the mesh, if the sculpture is outside the field of view of the sensor due to a wrong flight path, etc.

Furthermore, the size of the volume which can be reconstructed with the PCL Kinect Fusion is constrained to a limited size (i.e., 512 x 512 x 512 voxels), which was not enough to include the whole statue in a single flight scan. This, together with aforementioned issues, forced us to restart the reconstruction several times for different pieces of the statue, and acquire several meshes that were later aligned manually.

Robustness is an acknowledged issue with real-time reconstruction algorithms and rely mostly on the movement of the sensor around the scanned object. We found that also visualizing the current state of the reconstruction is of utmost importance. In our test case, this interaction is given by the state of the reconstruction and necessary sensor movement, and it is even more difficult since the movement is done remotely from the ground while the multicopter is several meters up in the air.

3.2. Real-time reconstruction with Jetson K1

In the second experiment, we tested the NVIDIA Jetson K1 board capabilities for real-time reconstruction. We did not mount this system on a multicopter yet since the reconstruction performance requires improvement. Here we present our current promising results on this platform. The Jetson K1 is a parallel mobile platform which has been recently released by Nvidia for embedded applications such as computer vision, robotics, and automotive [Teg]. It delivers 326 GFlops at low cost ($192) and low power consumption (10w). The Jetson TK1 is powered by Tegra K1, which currently sets the performance record of embedded system chips. the hardware set-up is depicted in Fig. 5.

We compiled the PCL Kinect Fusion implementation for the ARM platform, and then analyzed its performance using the

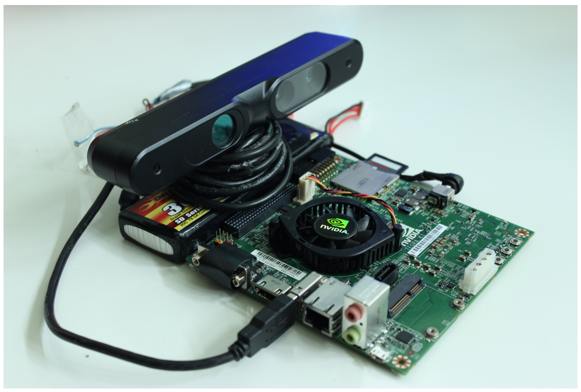
Figure 5: Jetson board connected to the ASUS Xtion depth sensor on top of a battery

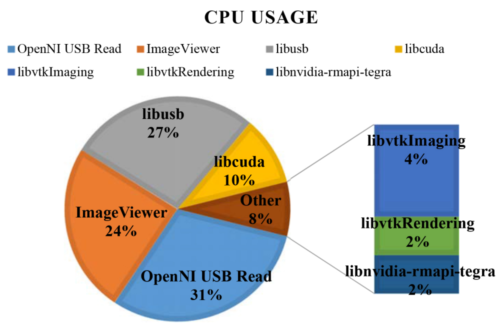

Figure 6: Experiment 2 library calls and their CPU usage

NVIDIA System Profiler. The obtained average frame rate is 3.1 fps while moving the sensor and 4.1 fps while keeping the sensor still. Running the application from the shell only and removing the windows that render the reconstruction and the depth maps yield an average of 4.2 fps when moving the sensor and 4.9-5.2 while still. This frame rate was considered not sufficient in order to try the reconstruction on the multicopter, so we started to analyze bottlenecks with NVIDIA System Profiler. Fig. 6 shows the library calls responsible for the highest CPU usage together with the relative percentage of CPU load within a single frame.

Reading from the USB requires the highest CPU load (`xnUSBReadThreadMain`: 22%, `libusb`: 19.7%). Regarding the CUDA kernel calls, the following have the highest call frequency: `combinedKernel` (42.7%), `tsdf23` (15.7%), `computeNmapKernel` (13.3%), `rayCastKernel` (11.2%), `bilateralKernel` (10.8%).

Based on our tests, additional factors that influence the quality of the reconstruction are:

- *the angle between the reconstructed surface and the camera*: if the angle is perpendicular on the reconstructed surface, then the quality of the mesh is visibly higher.
- *the distance from the sensor to the surface*: a closer prox-

Dev	File	Argument	Description
FS	`tune2fs`	`-c 1 /dev/mmcblkp02`	Check always
FS	`/etc/default/rcS`	`FSCKFIX=yes`	Autocorrection
SW	`/etc/init.d/rc.local`	`swapoff -a`	Turn off swap
CPU	`/boot/config.txt`	`force_turbo=1`	Turn off dynamic freq.

Table 1: Experiment 1, parameters for filesystem (FS), swap (SW), CPU

imity to the reconstructed surface results in a higher quality mesh.
- *the path of the sensor*: the path of the sensor movement around the scanned statue reduce the holes in the mesh geometry. At the object boundaries, the sensor should be moved so that the object is completely scanned.
- *planar surfaces often result in failure of tracking of the camera position.*

4. Discussion and future work

We presented two experimental setups to obtain the geometry of a statue that does not allow a direct scan from the ground. The first experiment employed the widely available Raspberry Pi platform. The reconstruction process and results showed the need for visualizing and processing the depth maps in real-time.

Concerning RQ1, and based on the insights from our first set-up (Sec. 3.1), we developed the hardware set-up using the the Jetson K1 which enables 3D reconstruction in real-time on the multicopter (Sec. 3.2). This allowed for the online analysis of the current state of the reconstructed mesh, enabling its correction and re-scanning if necessary. In this second experiment we show that nearly real-time reconstruction is achievable, however faster performance is desirable and more optimization is required. Also, this set-up posed new research questions, such as:

- what is a good path for moving the sensor?
- what interface and visualization would allow the user to have a complete understanding of what was scanned?
- how shall the path of scanning be modified so that the user is able to correct the scan?

Concerning RQ2, we found that real-time reconstruction and visualization is necessary in order to fully reconstruct and obtain a quality mesh of the real object. Controlling the octocopter flight path and checking the state of the reconstruction is a challenge, but might be solved through a screen that is permanently in the field of view of the user who controls the octocopter.

Future work could include visualization and analysis of the reconstruction state in real-time and to explore the interaction of the user while in the loop of this data-intensive process. Future applications could include prototyping non-wearable applications for interaction in motion consisting in projecting interfaces from above around the body of users, similar to the Autonomous Wandering Interface concept [Luk14]. Such mobile platforms would follow users and augment the physical space around them with contextual information. The work presented here is the basis for developing interaction techniques and visualization methods relying on real-time reconstruction methods.

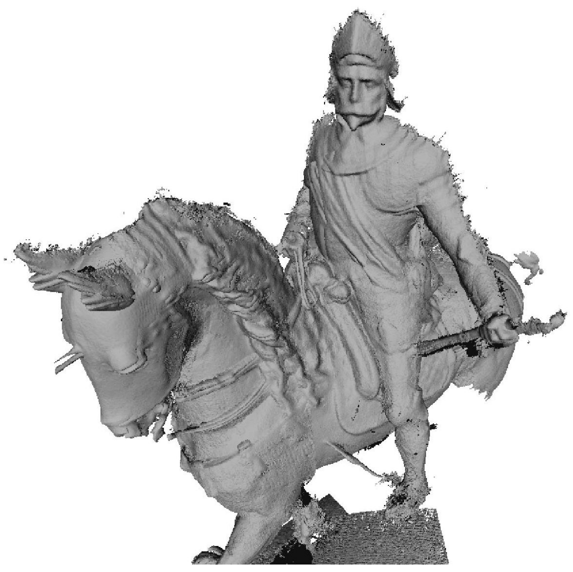
Figure 7: Reconstructed 3D model of the statue.

5. Acknowledgments

This work was supported by the EU FP7 People Programme (Marie Curie Actions) under REA Grant Agreement 290227.

References

[AMT*12] ALDOMA A., MARTON Z.-C., TOMBARI F., WOHLKINGER W., POTTHAST C., ZEISL B., RUSU R. B., GEDIKLI S., VINCZE M.: Point cloud library. *IEEE Robotics & Automation Magazine 1070*, 9932/12 (2012). 2

[BBBE14] BERTRAMA T., BOCKB T., BULGAKOVC A., EVGENOVD A.: Generation the 3d model building by using the quadcopter. *The 31st International Symposium on Automation and Robotics in Construction and Mining (ISARC 2014)* (2014). 1, 2

[BBR12] Bry A., Bachrach A., Roy N.: State estimation for aggressive flight in gps-denied environments using onboard sensing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on* (2012), IEEE, pp. 1–8. 1

[BIS09] Buehler M., Iagnemma K., Singh S.: *The DARPA Urban Challenge: Autonomous vehicles in city traffic*, vol. 56. springer, 2009. 1

[BPHR11] Bachrach A., Prentice S., He R., Roy N.: Range - robust autonomous navigation in gps-denied environments. *Journal of Field Robotics 28*, 5 (September 2011), 644–666. 1

[CCC*08] Cignoni P., Callieri M., Corsini M., Dellepiane M., Ganovelli F., Ranzuglia G.: Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference 2008, Salerno, Italy, 2008* (2008), pp. 129–136. 3

[CCD*11] Callieri M., Chica A., Dellepiane M., Besora I., Corsini M., Moyés J., Ranzuglia G., Scopigno R., Brunet P.: Multiscale acquisition and presentation of very large artifacts: The case of portalada. *J. Comput. Cult. Herit. 3*, 4 (Apr. 2011), 14:1–14:20. 1

[CLH*14] Chow J. C., Lichti D. D., Hol J. D., Bellusci G., Luinge H.: Imu and multiple rgb-d camera fusion for assisting indoor stop-and-go 3d terrestrial laser scanning. *Robotics 3*, 3 (2014), 247–280. 2

[droa] Amazon primeair. `http://www.amazon.com/b?node=8037720011`. Accessed: 2015-07-30. 1

[drob] Swiss post, swiss worldcargo and matternet drone tests. `https://goo.gl/GcCu0o`. Accessed: 2015-07-30. 1

[ESC14] Engel J., Sturm J., Cremers D.: Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems (RAS) 62*, 11 (2014), 1646–Ú1656. 1

[Fal13] Falahati S.: *OpenNI Cookbook*. Packt Publishing, 2013. 3

[FHH*12] Fraundorfer F., Heng L., Honegger D., Lee G. H., Meier L., Tanskanen P., Pollefeys M.: Vision-based autonomous mapping and exploration using a quadrotor mav. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* (2012), IEEE, pp. 4557–4564. 2

[GJRVF*13] Gonzalez-Jorge H., Riveiro B., Vazquez-Fernandez E., Martínez-Sánchez J., Arias P.: Metrological evaluation of microsoft kinect and asus xtion sensors. *Measurement 46*, 6 (2013), 1800–1806. 3

[HHL*14] Heng L., Honegger D., Lee G. H., Meier L., Tanskanen P., Fraundorfer F., Pollefeys M.: Autonomous visual mapping and exploration with a micro aerial vehicle. *Journal of Field Robotics 31*, 4 (2014), 654–675. 2

[Hub14] Huber G.: *Full autonomous quadcopter for 3D reconstruction without external sensors*. Master's thesis, Linz University, `http://goo.gl/HGoi6A`, 2014. 2

[icr] Icri cities. `http://www.cities.io`. Accessed: 2015-07-30. 2

[IKH*11] Izadi S., Kim D., Hilliges O., Molyneaux D., Newcombe R., Kohli P., Shotton J., Hodges S., Freeman D., Davison A., Fitzgibbon A.: Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2011), UIST '11, ACM, pp. 559–568. 2

[LMT07] Levinson J., Montemerlo M., Thrun S.: Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems* (2007), vol. 4, Citeseer, p. 1. 1

[Luk14] Luke Vink, Jessica Cauchard, James A. Landay: Autonomous wandering interface (awi) – concept video. *`https://youtu.be/cqU_hR2_ILU`* (2014). 1, 5

[MFE13] Murru G., Fratarcangeli M., Empler T.: Practical augmented visualization on handheld devices for cultural heritage. In *Computer Graphics, Visualization and Computer Vision* (2013), Agency V. S.-U., (Ed.). 2

[mks] Mk sweden. `http://www.mksweden.se/`. Accessed: 2015-07-30. 3

[NDI*11] Newcombe R. A., Davison A. J., Izadi S., Kohli P., Hilliges O., Shotton J., Molyneaux D., Hodges S., Kim D., Fitzgibbon A.: Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on* (2011), IEEE, pp. 127–136. 2

[PCGS15] Pingi P., Corsini M., Ganovelli F., Scopigno R.: Fast and simple automatic alignment of large sets of range maps. *Computers & Graphics 47* (2015), 78 – 88. 2

[rec] Autodesk recap 360. `https://recap360.autodesk.com`. Accessed: 2015-07-30. 2

[RWFP06] Ratti C., Williams S., Frenchman D., Pulselli R.: Mobile landscapes: using location data from cell phones for urban analysis. *Environment and Planning B Planning and Design 33*, 5 (2006), 727. 2

[SAD*14] Scaramuzza D., Achtelik M. C., Doitsidis L., Friedrich F., Kosmatopoulos E., Martinelli A., Achtelik M. W., Chli M., Chatzichristofis S., Kneip L., et al.: Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *Robotics & Automation Magazine, IEEE 21*, 3 (2014), 26–40. 2

[SAS*14] Schneegass S., Alt F., Scheible J., Schmidt A., Su H.: Midair displays: Exploring the concept of free-floating public displays. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2014), CHI EA '14, ACM, pp. 2035–2040. 1

[tan] Google tango. `https://www.google.com/atap`. Accessed: 2015-07-30. 2

[Teg] Nvidia embedded computing. `https://developer.nvidia.com/embedded-computing`. Accessed: 2015-07-30. 4

[UH14] Upton E., Halfacree G.: *Raspberry Pi user guide*. John Wiley & Sons, 2014. 2, 3

[Zha12] Zhang Z.: Microsoft kinect sensor and its effect. *MultiMedia, IEEE 19*, 2 (2012), 4–10. 3