2019 EC³
2019 European Conference on Computing in Construction
Chania, Crete, Greece
July 10-12, 2019

A TOOLCHAIN FOR AUTOMATED ACQUISITION AND PROCESSING OF AS-BUILT DATA WITH AUTONOMOUS UAVs

Henk Freimuth[1], Markus König[1]
[1]Chair of Computing in Engineering, Ruhr-University Bochum, Germany

Abstract

BIM-driven projects depend on structured as-built information with high availability, but the acquisition process for visual data involves too much manual labour. Apart from acquisition, the generation of data with laser scanners and other methods requires computationally intensive registration processes. Unmanned Aerial Vehicles (UAVs) have gained attention for all applications that involve remote sensing, but still require piloting personnel. UAVs with newly developed reactive autonomy, guided by stereo cameras in addition to standard GPS, enable both automated operation and, more importantly, the generation of structured data that is aligned with the model. This work proposes a toolchain for producing structured point clouds of construction elements in progress.

Introduction

Building Information Modeling (BIM) and its temporal variant 4D-BIM are the conceptual foundation of a standardised, model-based construction planning and management. Planning staff can only make informed decisions on the basis of rich as-built information. The collection of information on the current state of construction projects is a vital aspect of this paradigm (Bosché et al. 2014, Xiong et al. 2013). The need for rich data and a high level of availability correlates with the complexity of large building projects. Therefore, the costs for maintaining timely and comprehensive as-built information increase in larger construction projects. As a consequence, any decision and judgement depends on a model that provides accurate, timely and insightful information. Apart from quantitative and qualitative facts, such as delivery dates, delays, volumes, and progress notifications from subcontractors, visual data provides the highest degree of information to a human operator. Not only should as-built information be insightful to the human viewer, but more importantly, offer information for spatial analysis and automated comparision with as-planned model data. An ideal method for comparing as-planned and as-built status is able to determine the state of distinct BIM objects and present it in a meaningful way. An automated information generation workflow is a premise for automated progress reports and highlighting of issues per process. With the availability of photogrammetry and terrestial laser scanners (TLS), point clouds have become an invaluable data structure for measuring as-built status. Point cloud data is both the basis for meaningful 3D visualisation and moreover an adequate data structure for spatial analysis of BIM objects. However, TLS- and imaging-based approaches involve substantial manual labour, and therefore (multiple) daily updates are not feasible without disturbing ongoing processes and unreasonable amounts of man hours (Bosché et al. 2014).

Unmanned Aerial Vehicles (UAV), often referred to as drones, have recently gained attention for their ability to quickly gather information, predominantly photographs, especially in locations hard to reach. UAV is a broad term that applies over a wide range of aircraft, from winged plane-like types with several metres wingspan to helicopters that fit in the palm of a hand. Multirotors, the kind of UAV referred to in this work, are the preferred form factor for applications that require stable flight behaviour, vertical takeoff and landing, ability to carry payloads, and hovering in place for steady sensing. Research efforts on employing UAVs for monitoring (Eschmann et al. 2012), quality control (Wang et al. 2015, Morgenthal & Hallermann 2014), and digital reconstruction (Wefelscheid et al. 2011) leverage these advantages. Apart from enabling better and quicker remote sensing, the real advantage of UAV-based sensing lies in the potential of autonomous operation. Autonomy denotes a breaking point at which an effort to gather as-built information is not anymore measured per scanning operation, but instead is only necessary once for an arbitrary number of subsequent operations. Therefore, the UAV-based approach has the potential to make as-built measurements more efficient, with the scan rate possibly being increased from weekly to multiple times a day. Few examples of UAVs for inspection purposes have made practical use of autonomous capabilities in terms of avoiding obstacles and automatic flight towards inspection targets. Recent developments in the Robot Operating System (ROS) and the PX4 flight stack, however, reveal robust autonomous functionality for UAVs. The software for autonomous navigation used in this project has been proven to work on real UAVs and encourages a development workflow with simulator tests prior to outdoor flight tests. An open source ecosystem of robotics software is the foundation for this research work, incorporating the advantages of autonomous flight, state-of-the-art sensor technology, and BIM-driven workflows as the data model for detecting change in structural construction.

Prior to this work the authors investigated the feasibility of using UAVs with pre-computed flight missions to capture

DOI:10.35490/EC3.2019.137

photographs on construction sites. The UAV's navigation capabilities were limited by using only GPS for localisation purposes.

Contribution

This research work proposes a toolchain for as-built data generation, making use of autonomous UAVs. The toolchain is a combination of existing robotics tools and new ones, developed in the course of this project. In combination, these tools enable a UAV to automatically capture visual information of structural elements on the level of BIM objects. This work is part of an ongoing research of employing UAVs for inspection and monitoring purposes. The toolchain is under active development, as is the underlying autopilot and robotics codebase. Following this stage of development and simulator tests, this project will reach an evaluation stage with practical case studies. This work continues to employ the open source PX4 flight stack which was used in prior work (Freimuth & König 2018), but replaces GPS navigation, which is unaware of its environment, with reactive autonomy. Vision-based environment sensing, implemented in ROS, is used for safe navigation and path planning in an environment that is partially known. The toolchain that constitutes the autonomous UAV and the automated as-built data generator is a combination of modules, developed specifically for path-planning on BIM and scanning construction objects, based on a stack of open source robotic modules from the ROS and PX4 software projects. The known environment is derived from BIM, either as an as-planned model for a specific date and time or as an as-built model from a previous scan. The point cloud processing toolchain, implemented in the course of this research, produces point clouds with colour information that are aligned with the underlying BIM. Furthermore, by eliminating all points of objects known from the previous state, the point clouds produced by the proposed method represent only new or modified objects. Effectively, the toolchain substantially reduces the amounts of data to store and provides a per-object view on the deviations between as-planned and as-built. The resulting object point clouds represent distinct groups of points that belong to semantic objects, which is beneficial to further processing methods from the scan-to-BIM domain.

Research Background

3D information of existing structures is highly anticipated. The data is required for progress tracking (Son et al. 2015, Bosché et al. 2015, 2014), structural health monitoring (Klein et al. 2012, Jiang et al. 2008), quality assessment (Chen & Luo 2014, Tang et al. 2011, Kim et al. 2015) and as-built modeling (Tang et al. 2010). The importance of efficient measuring methods becomes evident, as each of these applications depends on as-built data input and can only be executed as often as new data can be generated. Huber et al. (2011) describe the process of creating as-built data as time-consuming and error-prone and one of the key barriers to the widespread use of as-built BIMs in industry.

Taking photographs of objects for state documentation purposes is ubiquitous. However, taking images for structured analysis and as-built BIM purposes imposes special requirements. There are two major approaches for acquiring the state of a construction project with photographs. The first method takes images of objects and registers them by estimating the camera pose, with which the image was taken. This camera pose is then applied to a virtual camera inside the corresponding 3D model space, or from a 4D model at investigation time. A comparison of the virtual image and the real one then exposes differences between model and reality. By applying computer vision techniques, features in images or differences between the as-built and the as-planned images may be identified (Lukins & Trucco 2007, Golparvar-Fard et al. 2009, Ibrahim et al. 2009). Such techniques can produce various results like qualitative statements about surface properties (e.g. cracks in concrete slabs), detection of BIM objects (e.g. image contains slab), and clash detection (e.g. pipes cannot be installed as planned; clash with cable harness). The second method applies Structure from Motion (SfM) to derive 3D geometry in the form of dense point clouds. Dozens of images must be taken for each object and the SfM process itself takes several hours, even when using specialised commercial software and GPU-acceleration. Golparvar-Fard Mani et al. (2015) present a machine learning-based plane matching with Support Vector Machines (SVM) that relies on SfM. Klein et al. (2012) conducted a study on generating exterior and interior as-built models with photogrammetry methods. The authors acknowledge limitations regarding measurements of outer structures in upper stories, a limitation that applies to all inspection methods carried out by surveyors.

Terrestial Laser Scanning (TLS) is an alternative method for generating 3D geometry. TLS is generally more accurate than image-based reconstruction, mostly due to the implicit error of the triangulation approach of SfM (Tang et al. 2010, Wilkinson et al. 2016). However, conventional TLS only provides 3D points without colour information. Other approaches combine TLS and camera systems, which provide colour by mapping pixel values to points of the laser scan. TLS devices need to be mounted stationary for each scan, during which non-occluded surfaces will be captured. As a consequence, partially occluded and free-standing objects require multiple scans. When dealing with multiple scans of the same object, individual point clouds must be joined in a registration process. Automatic

registration can be achieved with heuristic approaches like the Iterative Closest Point (ICP) algorithm or other approaches like the 4-Points Congruent Sets method (Hichri et al. 2013, Theiler et al. 2014, Besl & McKay 1992).

Concluding the data acquisition methods, laser scanning and image-based approaches (e.g. photogrammetry) have greatly improved the measurement process over the manual method and mitigate the risk of modeling errors (Anil et al. 2013). However, despite producing richer data with improved accuracy, the sensors still require manual effort for each scanning task. Setup and operation of TLS- and camera-based inspections are time-consuming and cause interferences with other construction tasks (Tang et al. 2010). With data generated in an automated manner, the number of scans could be increased from weekly to daily intervals.

Regardless of the method, TLS or SfM, point cloud data has no inherent alignment in the model space. In the context of BIM-driven construction the sole collection of unordered 3D information is not sufficient. Markerless registration of as-built point cloud data with BIM mesh data is a necessity for automated progress monitoring. The aforementioned 4-Points Congruent Sets method was adapted as 4-Plane Congruent Set algorithm with robust results (Bueno et al. 2018).

After registration, a point cloud is still an unstructured mass of data that resembles a whole scene. In order to make semantic decisions, point clouds need segmentation, preferably on a structural object level. Various segmentation approaches for structural objects have been investigated. A colour-based segmentation coupled with region growing yielded good results, but depends on colour information and therefore also on suitable lighting conditions (Zhan et al. 2009). Approaches based on shape detection (Ning et al. 2009) yield good results for detection of surfaces, but fall short in the segmentation of semantic objects with individual forms. The Point Cloud Library (PCL) implements several methods for segmentation and clustering (Rusu & Cousins 2011), including colour-based and plane-based methods, which will be investigated in upcoming research work.

Concept

The objective of this paper is the development of a novel method for measuring the as-built status with autonomous UAVs. The integration of UAVs requires a technological bridge between model information and controlling the autonomous aircraft. Control in this context means being able to interface with the path planning unit of the UAV's flight control.

Choosing a suitable platform for UAV development

Common (commercial) UAV platforms allow for manual control and mission control, with missions being sequences of GPS waypoints. For the sake of usability these platforms offer mobile applications with limited functionality. The envisioned integration and automation however poses the requirement of full programmatic control over the aircraft. Some commercial manufacturers offer optional development programs in the form of Software Development Kits (SDK). Committing to a single manufacturer creates an unnecessary dependency towards the company that maintains the SDK, but lifespan and support may be limited. This work is based on PX4, an open source software and hardware platform for UAVs. It was originally started as a research platform for vision-based autonomy and became a robust UAV platform. PX4 gives developers full control over the UAV and extensive compatibility with the Robot Operating System (ROS). The PX4 firmware is under active development and testing by researchers, industrial partners, and enthusiasts. An active community of developers and testers provides feedback on proposed changes to the system and analysable flight logs. This makes for a significant advantage in quality control over many other platforms. The sub project *PX4/Avoidance* (2016) is another basis of this work, focused on reactive autonomy with path planning in known and unknown environments.

The Robot Operating System (ROS)

ROS was started in 2007 at the Stanford Artificial Intelligence Laboratory and became part of the Open Source Robotics Foundation (OSRF) in 2012. It is an open source framework for the development of and execution on actual robots. At the core of the ROS concept is a graph architecture, which considers all robot-related processes and entities as distinct nodes while messages passed among nodes are considered edges. Apart from this architectural aspect, ROS is also a large collection of established robotics modules that come from an international community. Abstraction of message types and networking communication allows single modules of robotic systems to be executed on different computer systems. Furthermore, the modular architecture makes nodes exchangeable under the premise of conformity with the required message protocol. This is an essential aspect for research purposes, as it allows developers to change certain functionality of a robot with little effort. Iterative changes in code need testing and verification, but conducting outdoor experiments for every test is impracticable. Malfunctioning code could destroy hardware or even put humans in danger, when testing flying robots. Therefore, simulation driven development and testing has become an indispensable requirement.

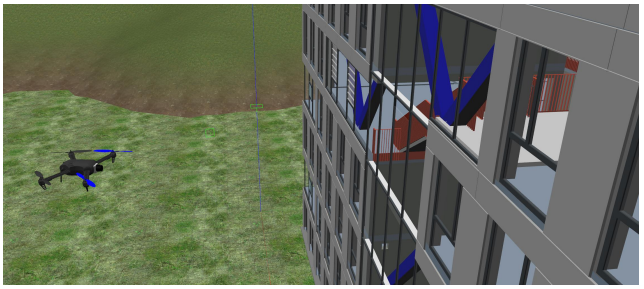
Simulation in robotics

Simulation plays a major role in the development of robots, especially with autonomous UAVs, where malfunction on the device causes crashes, destroying hardware and possibly causing harm to bystanders and objects. Outdoor experiments with untested features significantly slow down the development process and pose unwanted risks and costs. For these reasons the robotics community established a development method that employs realistic simulations up to a point of validation where real world experiments can be conducted (Visser et al. 2011, Symington et al. 2014, Olivares-Mendez et al. 2014). Gazebo was created for exactly this purpose and is a popular simulator with deep integration in ROS (Koenig & Howard 2004, Meyer et al. 2012). The simulator can be configured to use exchangeable phyiscs engines and a plugin system enables integration of new virtualised sensors. Within the simulation, the flight controller receives virtual sensor data including the UAV's attitude, axis rotation rates and acceleration. Instead of controlling actual motors, the flight controller feeds the simulation with actuator commands. The real benefit here is that experiments carried out in the simulation run the same code as the real UAV in outdoor usage. When transitioning from simulator experiments to outdoor execution, only the virtual sensors need to be replaced by their real counterparts.

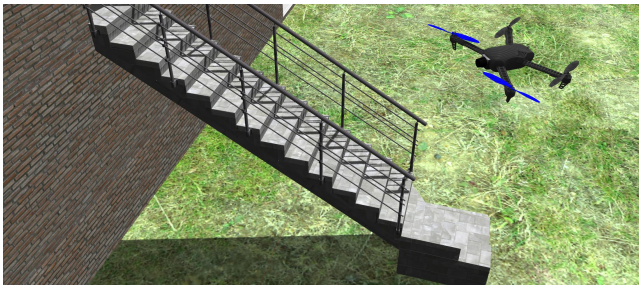
Simulating the UAV for automated as-built scans

In the scope of this work generation and processing of point cloud data is also based on simulation data. A camera plugin in Gazebo provides the toolchain with images from the perspective of the UAV's stereo camera. The simulation environments are defined in world files, which describe the composition and properties of physical objects including the ground profile, building structures, and robots.

Fig. 1 shows the simulated autonomous UAV navigating around two examples of building models. The building models represent the construction site at scan time. They were created by converting an IFC file to a Collada 3D asset file. Collada is a modern and open 3D asset exchange format which is maintained by the Khronos Group. The conversion is achieved with the IfcBlender tool (Krijnen 2011). This converted model is necessary for the simulation scene to resemble what the real construction site looks like. The definition of simulation scenes is straight forward, however one additional step was found necessary in the preparation of experiments. During the first trials with the vision-based flight stabilisation, the UAV behaved erratically when its cameras were oriented towards the building. It was found that some of the converted building models had only few visible surface features, which are needed for the stereo cameras to create a disparity



(a) Example of a building model with complex geometry.



(b) Example of a building with realistic textures. The additional surface features greatly improve the vision-based depth sensing.

Figure 1: The simulated UAV autonomously navigates around virtual BIM-derived building scenes.

map. The building models were augmented with realistic surface textures (i.e. the brick surface wall in Fig. 1b) which immediately restored the stable vision-based flight behaviour.

Gazebo provides a plugin to access virtual camera streams of the 3D scene. Simulated cameras are defined by various properties such as resolution, field of view, frame rate, distortion and noise. These parameters make the virtual cameras configurable to resemble the properties of real cameras. Furthermore they are linked with the simulated UAV model, providing pictures from its point of view. This allows working with synthetic visual sensors as input for robot perception.

Instead of using the common mission flight modes of PX4, a path planner node in ROS takes over control. The planner node is responsible for navigating the UAV towards its navigation goal. Currently, there are two instances of planning nodes available, *local_planner* and *global_planner*. The local planner is able to navigate in a previously unknown environment and find ways around obstacles that come up during its course. The local planner makes decisions based on the 3D Vector Field Histogram (3DVFH+). The 3DVFH+ is a computationally efficient two-dimensional array of potential motion directions that are either free or blocked. The elements of the array are projected on a polar sphere around the UAV and represent 3D motion vectors in all directions. The cell's values are updated in real-time

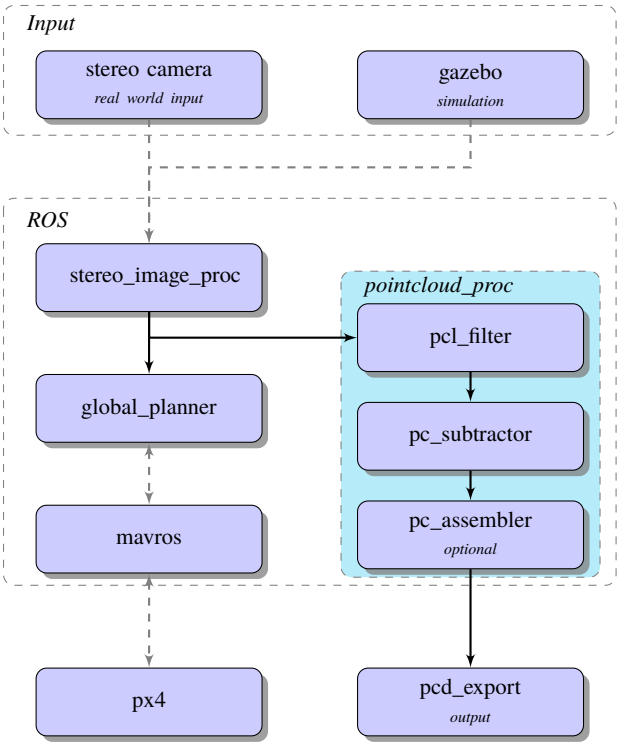


Figure 2: The component architecture of the proposed toolchain. Nodes represent individual software modules, arrows indicate data flow between the modules, with solid arrows marking the flow of point cloud data.

by determining if a certain path is blocked or free in the octomap (Vanneste et al. 2014). The *global_planner* additionally recognises a previously known enviroment and maintains a global representation of the previously known enviroment. Both variants of the planner continuously exchange messages with the *mavros* node, which is the communication interface between ROS and the PX4 instance that runs independently, for example on a real-time microcontroller instead of the development machine.

Components of the toolchain

The software architecture of the toolchain is in accordance with the modularised structure of ROS. It makes use of existing structures from the PX4/avoidance project with additional modules developed specifically for the purpose of measuring the as-built status of construction sites and BIM.

Fig. 2 provides an overview of the toolchain with an emphasis on the data flow between its modules. Each node in the graph is a software component that handles a certain task and will be presented individually in the following subsections. The relations between nodes depict which

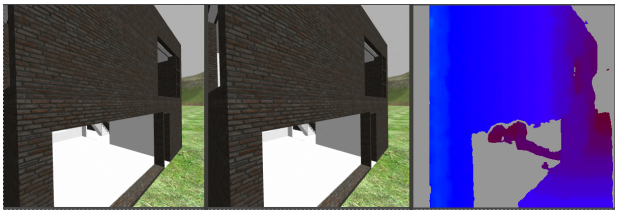
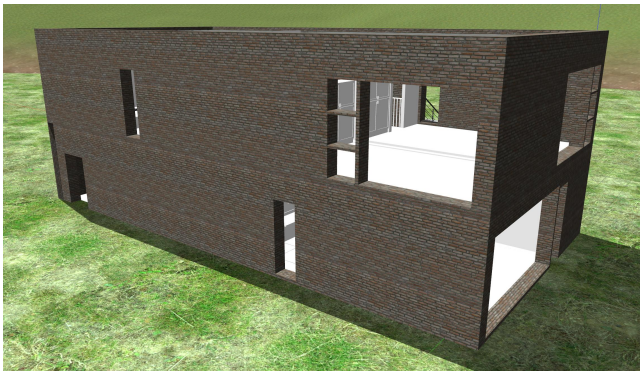


Figure 3: A visualisation of the generation of the disparity map (right) from the stereoscopic camera images (left). The color gradient in the disparity map represents perceived depth.

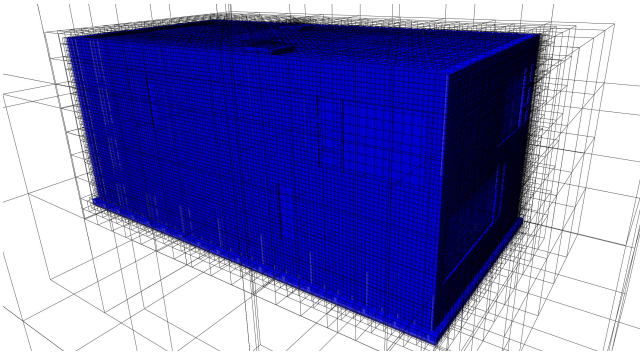
nodes exchange data with each other and therefore resemble dependencies. The top group *Input* includes nodes that generate the required stream of stereo images. This stream either comes from actual stereo cameras on the UAV (e.g. Intel RealSense), or from virtual cameras that adapt the real device by capturing images with comparable properties in the 3D simulation environment. This is the main input for the group of *ROS* nodes that process the stereo images for navigation and create the desired point clouds that represent the as-built status.

The *stereo_image_proc* node (included in the standard ROS distribution) is responsible for processing the stereo image stream. In its initial stage it computes a disparity image, in which each pixel value represents the depth of the same pixel in the rectified input image. Fig. 3 shows an example of the binocular camera image and the resulting colour-coded disparity map with a gradient between blue (near) and red (far). With depth information and color available, the node is further capable of producing a ROS typed `sensor_msgs::PointCloud2` message. `sensor_msgs::PointCloud2` is the default type for all further message passing of point cloud data and is compatible with the `pcl::PointCloud2` from the PointCloud Library, which offers mature point cloud processing methods. The flow of the resulting point cloud data splits at this point, as it is being used as input stream for both the planner node (local or global) and the nodes in the subgroup *pointcloud_proc*. Devices like Intel RealSense can compute disparity maps on their own. The accuracy and performance of these will be evaluated in upcoming work, but an integrated solution is expected to make the processing chain more CPU-friendly.

The subgroup *pointcloud_proc* encapsulates software modules for real-time processing of point cloud streams. The first notable node in this subgroup is *pcl_filter*, which is a PCL-based passthrough filter for eliminating all points outside a certain range in one dimension. In this case, the passthrough filter is used to eliminate all points near ground. Ground points are relevant to navigation applications, but since the objective of this toolchain is the mea-



(a) Example building in Gazebo



(b) Corresponding octree for navigation and point cloud processing

Figure 4: The simulated building scene and its corresponding octree, used on the UAV as a computationally efficient representation of navigation space.

surement of building structures, data of the ground is of no further interest. Such filtering processes significantly increase the performance of all subsequent operations. PCL implements further filtering routines. The statistical outlier removal is one notable filter in which point clouds can be freed from outlier points that do not represent solid structures. The next node in the processing flow is the *pc_subtractor* node.

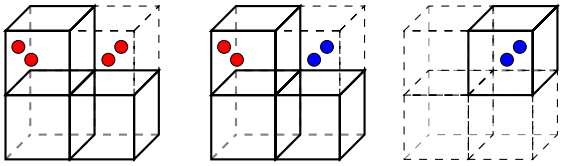
The point cloud subtractor

The *pc_subtractor* node is the second to last stage in the *pointcloud_proc* subgroup and is responsible for reducing and separating input point clouds into groups of points that belong to objects of interest. During each flight, the UAV captures extensive amounts of point cloud data. Subsequent point cloud messages are assembled at the end of the toolchain with the intention to create complete views on objects from all perspectives. Although the input rate can be throttled to a value lower than that of the navigation system, it should be fast enough to ensure that no objects are overlooked while the UAV is moving. The information of single frames, accumulated intervals of frames or even whole flight sequences is too large to make sense to a human operator when simply displayed on a screen. When considering a structured storage and processing of the data, it is also advisable to have point cloud data that is already clearly registered with individual objects in BIM. Therefore the inspection system automatically reduces the input data to solely represent new structures that weren't present during the last scan. The method requires accurate as-planned data for the automatic identification of new structures. This concept assumes that the data of each inspection is added to the existing as-planned data and therefore constitutes the as-planned model for the next inspection. In technical terms, the information on existing structures is extracted from BIM. In a 4D-BIM workflow this means that each inspection is executed on the current state of the model data and its result will be used to create the next iteration. Inside the *pc_subtractor* node the existing structures are processed as octree representations. The octree is generated directly from the mesh data of the BIM with the open source programs *binvox* (Min 2004, Nooruddin & Turk 2003) for voxelisation and *binvox2bt* (Hornung et al. 2013) for creating an octree file from the voxels.

Fig. 4 shows an example simulation with a building model derived from IFC and its corresponding octree representation. The building's octree is used as initial input to the dynamic octomap, which is constantly updated in flight.

Preloading the octomap with the known occupied space of the building model has a positive effect on the UAV's capability of finding a shortest viable path, since all known structures are automatically included in the path planning of the local planner. The octree is an adequate data structure to efficiently represent the spatial extent of geometric objects. The current environment is managed in Octomap (Hornung et al. 2013), an octree-based environment representation for robot navigation. Alignment and scaling of the input point clouds is a further requirement. The UAV inspection concept of this work makes use the autopilot's self localising capability, a key feature of the autonomous flight system. The point clouds that are used for obstacle detection and avoidance receive accurate transforms (translation, scale) from the autopilot that constantly signals its current position in space to the *mavros* node.

Fig. 5a shows the initial status of data as it arrives in the node. The existing structures are displayed as boxes with thick solid outlines and the input data is illustrated by red points. The boxes represent leafs of the octree, which is an efficient discretisation of the building geometry. Some points are placed within a box of the octree, others are outside the octree. Since the octree is a discretised representation of the as-planned model, all points inside the octree shall be discarded. The goal is to efficiently identify all points, that lie outside the octree. The *pc_subtractor* node



(a) input points; octree of existing structure (solid boxes)

(b) elimination process; blue points remain

(c) resulting set of points in new structure

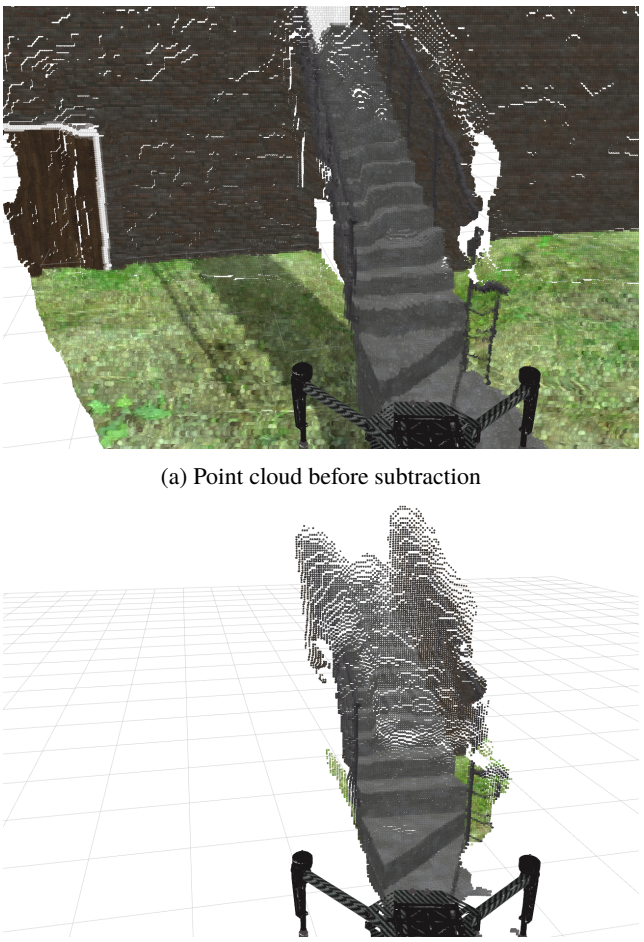
Figure 5: Illustration of the point cloud subtractor using the octree to subtract points that are outside of known structures.

makes use of the PCL data structure `OctreePointCloud` and its function `isVoxelOccupiedAtPoint` to test each input point within in the octree and keep a set of outlier points, marked as blue points in Fig. 5b, Fig. 5c.

The result of this operation is a point cloud frame that represents only objects that are new or modified compared to the previous model of as-built or as-planned status. Fig. 6a shows an example of a point cloud before processing in the subtraction node. In this case, the wall was part of the known structure and the staircase is not yet known to the as-built model. Fig. 6b shows the point cloud after processing. The points in the ground region were removed by the passthrough filter and the subtractor node removed the wall of the building. The expected result of this process is a point cloud with an isolated representation of the staircase. This isolated representation is easy to comprehend at a first glance. Furthermore, the size of the point cloud in storage is significantly smaller with similar benefits for subsequent processing operations. Most importantly though, the point cloud is semantic as it only represents one object. This is an advantage for further operations like reconstruction of meshes from point clouds. The *pc_subtractor* node is followed by the *pc_assembler* node, used for combining subsequent frames of point clouds.

The point cloud assembler node

Each point cloud frame, regardless of it being generated by a TLS or a flying stereo camera, can only represent points visible from the location of the sensor. For 3D mesh reconstruction purposes, point clouds of objects should be complete. In order to create complete point clouds of physical objects, point clouds must be created from multiple perspectives and then these frames must be aligned and joined. The autonomously flying UAV continuously captures point cloud frames while navigating around construction objects. The *pc_assembler* node collects a certain number of frames together with their respective transforms, an ROS description of pose, orientation, and scale of objects. It takes care of joining those frames and outputs a single, accumulated point cloud. The last step is



(a) Point cloud before subtraction

(b) Point cloud after subtraction of points

Figure 6: Visualisation of the subtraction operation on point clouds. The points of the staircase remain, ground and facade points are eliminated.

the *pcd_export* node, which receives point cloud messages and writes artifact files in the common PCD file type.

Observations

The Gazebo simulation application loads meshes for arbitrary objects from Collada files. This also applies to the geometric representation of the known building elements, which are derived from an Industry Foundation Classes (IFC) file to Collada format. The model files are converted with the IfcBlender plugin for Blender (Krijnen 2011). Initial tests on converted building models yielded poor results due to the plain surfaces of the Collada exports. The material properties of IFC objects have no realistic visual representations in the form of textures. Visible surfaces are defined by colors that will also define the material descriptions of the Collada assets. The resulting surfaces were found to have negative effects on

the depth perception in the stereo image proc node, which relies on visual surface features to compute a meaningful disparity map. As a result the simulated UAV would not be able to navigate properly. The problem was solved by adding photorealistic pattern textures of bricks and gravel stone to facade elements. This is a limitation that only applies to the simulation. Camera input from physical objects under natural lighting exposes more visual features than a simulation.

Conclusions and Outlook

This work is a study on automating and improving the process of as-built data generation. The proposed toolchain is capable of automatically guiding a UAV around building structures, effectively enabling it to capture as-built info on an object level. By automating the process of data acquisition, the toolchain effectively reduces repetitive manual labour which is needed for collecting visual data of structural elements. The self-localising UAV significantly improves the process, as scanned point clouds can inherit the UAV's position information for each frame. Therefore, no additional point cloud registration is needed. Furthermore, the toolchain's ability to eliminate all points of known or unmodified objects greatly improves the value of the as-built data. Despite being under active development, the open source autopilot software and the Robot Operating System provide a solid foundation for practical applications like the one presented here.

This is a work in progress. Following the simulation-based tests, the toolchain will be evaluated on real hardware and in practical case studies. The isolated groups of new or modified objects will be used for automated progress monitoring of processes in 4D-BIM.

References

Anil, E. B., Tang, P., Akinci, B. & Huber, D. (2013), 'Deviation analysis method for the assessment of the quality of the as-is Building Information Models generated from point cloud data', *Automation in Construction* **35**, 507–516.

Besl, P. J. & McKay, N. D. (1992), 'A method for registration of 3-D shapes', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2), 239–256.

Bosché, F., Adrien, G., Yelda, T., Haas, C. T. & Haas, R. (2014), 'Tracking the Built Status of MEP Works: Assessing the Value of a Scan-vs-BIM System', *Journal of Computing in Civil Engineering* **28**(4), 05014004. -.

Bosché, F., Ahmed, M., Turkan, Y., Haas, C. T. & Haas, R. (2015), 'The value of integrating Scan-to-BIM and Scan-vs-BIM techniques for construction monitoring using laser scanning and BIM: The case of cylindrical MEP components', *Automation in Construction* **49**, 201–213.

Bueno, M., Bosché, F., González-Jorge, H., Martínez-Sánchez, J. & Arias, P. (2018), '4-Plane congruent sets for automatic registration of as-is 3D point clouds with 3D BIM models', *Automation in Construction* **89**, 120–134.

Chen, L. & Luo, H. (2014), 'A BIM-based construction quality management model and its applications', *Automation in Construction* **46**, 64–73.

Eschmann, C., Kuo, C. M., Kuo, C. H. & Boller, C. (2012), Unmanned aircraft systems for remote building inspection and monitoring, *in* '6th European Workshop on Structural Health Monitoring', pp. 1–8.

Freimuth, H. & König, M. (2018), 'Planning and executing construction inspections with unmanned aerial vehicles', *Automation in Construction* **96**, 540–553.

Golparvar-Fard, M., Peña-Mora, F. & Savarese, S. (2009), 'Application of D4AR A 4-Dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication', *Journal of Information Technology in Construction (ITcon)* **14**(13), 129–153. `http://www.itcon.org/2009/13`.

Golparvar-Fard Mani, Peña-Mora Feniosky & Savarese Silvio (2015), 'Automated Progress Monitoring Using Unordered Daily Construction Photographs and IFC-Based Building Information Models', *Journal of Computing in Civil Engineering* **29**(1), 04014025.

Hichri, N., Stefani, C., Luca, L. D., Veron, P. & Hamon, G. (2013), From point cloud to BIM: A survey of existing approaches, *in* 'XXIV International CIPA Symposium', Proceedings of the XXIV International CIPA Symposium, p. na.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. & Burgard, W. (2013), 'OctoMap: An efficient probabilistic 3D mapping framework based on octrees', *Autonomous Robots* **34**(3), 189–206.

Huber, D., Akinci, B., Anil, E., Okorn, B. E. & Xiong, X. (2011), Methods for automatically modeling and representing as-built building information models, *in* 'Proceedings of the NSF CMMI Research Innovation Conference', Atlanta, Georgia.

Ibrahim, Y. M., Lukins, T. C., Zhang, X., Trucco, E. & Kaka, A. P. (2009), 'Towards automated progress assessment of workpackage components in construction projects using computer vision', *Advanced Engineering Informatics* **23**(1), 93–103.

Jiang, R., Jáuregui, D. V. & White, K. R. (2008), 'Close-range photogrammetry applications in bridge measurement: Literature review', *Measurement* **41**(8), 823–834.

Kim, M.-K., Cheng, J. C. P., Sohn, H. & Chang, C.-C. (2015), 'A framework for dimensional and surface quality assessment of precast concrete elements using BIM and 3D laser scanning', *Automation in Construction* **49**, 225–238.

Klein, L., Li, N. & Becerik-Gerber, B. (2012), 'Imaged-based verification of as-built documentation of operational buildings', *Automation in Construction* **21**, 161–171.

Koenig, N. & Howard, A. (2004), Design and use paradigms for Gazebo, an open-source multi-robot simulator, *in* '2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)', Vol. 3, pp. 2149–2154 vol.3.

Krijnen, T. (2011), 'IfcOpenShell', `http://ifcopenshell.org/ifcblender.html`.

Lukins, T. C. & Trucco, E. (2007), Towards Automated Visual Assessment of Progress in Construction Projects., *in* 'Procedings of the British Machine Vision Conference 2007', pp. 1–10.

Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U. & von Stryk, O. (2012), Comprehensive Simulation of Quadrotor UAVs Using ROS and Gazebo, *in* I. Noda, N. Ando, D. Brugali & J. J. Kuffner, eds, 'Simulation, Modeling, and Programming for Autonomous Robots', Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 400–411.

Min, P. (2004), 'Binvox', `http://www.patrickmin.com/binvox`.

Morgenthal, G. & Hallermann, N. (2014), 'Quality Assessment of Unmanned Aerial Vehicle (UAV) Based Visual Inspection of Structures', *Advances in Structural Engineering* **17**(3), 289–302.

Ning, X., Zhang, X., Wang, Y. & Jaeger, M. (2009), Segmentation of Architecture Shape Information from 3D Point Cloud, *in* 'Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry', VRCAI '09, ACM, New York, NY, USA, pp. 127–132.

Nooruddin, F. S. & Turk, G. (2003), 'Simplification and Repair of Polygonal Models Using Volumetric Techniques', *IEEE Transactions on Visualization and Computer Graphics* **9**(2), 191–205.

Olivares-Mendez, M. A., Kannan, S. & Voos, H. (2014), Setting up a testbed for UAV vision based control using V-REP amp;amp; ROS: A case study on aerial visual inspection, *in* '2014 International Conference on Unmanned Aircraft Systems (ICUAS)', pp. 447–458.

PX4/Avoidance (2016), `https://github.com/PX4/avoidance`.

Rusu, R. B. & Cousins, S. (2011), 3D is here: Point Cloud Library (PCL), *in* '2011 IEEE International Conference on Robotics and Automation', pp. 1–4.

Son, H., Bosché, F. & Kim, C. (2015), 'As-built data acquisition and its use in production monitoring and automated layout of civil infrastructure: A survey', *Advanced Engineering Informatics* **29**(2), 172–183.

Symington, A., Nardi, R. D., Julier, S. & Hailes, S. (2014), Simulating quadrotor UAVs in outdoor scenarios, *in* '2014 IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3382–3388.

Tang, P., Anil, E. B., Akinci, B. & Huber, D. (2011), Efficient and Effective Quality Assessment of As-Is Building Information Models and 3D Laser-Scanned Data, *in* 'Computing in Civil Engineering (2011)', American Society of Civil Engineers, Miami, Florida, United States, pp. 486–493.

Tang, P., Huber, D., Akinci, B., Lipman, R. & Lytle, A. (2010), 'Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques', *Automation in Construction* **19**(7), 829–843.

Theiler, P. W., Wegner, J. D. & Schindler, K. (2014), 'Keypoint-based 4-Points Congruent Sets – Automated marker-less registration of laser scans', *ISPRS Journal of Photogrammetry and Remote Sensing* **96**, 149–163.

Vanneste, S., Bellekens, B. & Weyn, M. (2014), 3 DVFH + : Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap, pp. 89–100. `http://ceur-ws.org/Vol-1319/#morse14_paper_08`.

Visser, A., Dijkshoorn, N., van der Veen, M. & Jurriaans, R. (2011), Closing the gap between simulation and reality in the sensor and motion models of an autonomous AR.Drone, *in* 'Proceedings of the International Micro Air Vehicles Conference 2011 Summer Edition', p. 9.

Wang, J., Sun, W., Shou, W., Wang, X., Wu, C., Chong, H.-Y., Liu, Y. & Sun, C. (2015), 'Integrating BIM and LiDAR for Real-Time Construction Quality Control', *Journal of Intelligent & Robotic Systems* **79**(3), 417–432.

Wefelscheid, C., Hänsch, R. & Hellwich, O. (2011), 'Three-Dimensional Building Reconstruction Using Images Obtained by Unmanned Aerial Vehicles', *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **3822**, 183–188.

Wilkinson, M., Jones, R., Woods, C., Gilment, S., McCaffrey, K., Kokkalas, S. & Long, J. (2016), 'A comparison of terrestrial laser scanning and structure-from-motion photogrammetry as methods for digital outcrop acquisition', *Geosphere* **12**(6), 1865–1880.

Xiong, X., Adan, A., Akinci, B. & Huber, D. (2013), 'Automatic creation of semantically rich 3D building models from laser scanner data', *Automation in Construction* **31**, 325–337.

Zhan, Q., Liang, Y. & Xiao, Y. (2009), 'Color-based segmentation of point clouds', *Laser scanning* **38**(3), 155–161. http://www.isprs.org/proceedings/xxxviii/3-w8/papers/248_laserscanning09.pdf.