Towards Autonomous Robotic In-Situ Assembly on Unstructured Construction Sites Using Monocular Vision

C. Feng[a], Y. Xiao[a], A. Willette[b], W. McGee[b], and V.R. Kamat[a]

[a]Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, USA
[b]College of Architecture and Urban Planning, University of Michigan, Ann Arbor, USA
E-mail: cforrest@umich.edu, yongxiao@umich.edu, willetta@umich.edu, wesmcgee@umich.edu, vkamat@umich.edu

Abstract -

Unlike robotics in the manufacturing industry, on-site construction robotics has to consider and address two unique challenges: 1) the rugged, evolving, and unstructured environment of typical work sites; 2) the reversed spatial relationship between the product and the manipulator, i.e. the manipulator has to travel to and localize itself at the work face, rather than a partially complete product arriving at an anchored manipulator. The presented research designed and implemented algorithms that address these challenges and enable autonomous robotic assembly of freeform modular structures on construction sites. Building on the authors' previous work in computer-vision-based pose estimation, the designed algorithms enable a mobile robotic manipulator to: 1) autonomously identify and grasp prismatic building components (e.g., bricks, blocks) that are typically non-unique and arbitrarily stored on-site; and 2) assemble these components into pre-designed modular structures. The algorithms use a single camera and a visual marker-based metrology to rapidly establish local reference frames and to detect staged building components. Based on the design of the structure being assembled, the algorithms automatically determine the assembly sequence. Implemented using a 7-axis KUKA KR100 robotic manipulator, the presented robotic system has successfully assembled various structures autonomously as shown in Figure 1, demonstrating the designed algorithms' effectiveness in autonomous on-site construction robotics applications.

Keywords -

On-Site Construction Robotics; Autonomous Assembly; Modular Construction; Pose Estimation

1 Introduction

Several studies have argued that among all industries, construction has seen a significant productivity decrease over the last several decades compared to other industries [1]. Construction has also been documented to have some of the highest rates of workspace injuries and fatalities [2]. Automation and robotics in construction (ARC) has the potential to relieve human workers from repetitive and dangerous tasks, and has been extensively promoted in the literature as a means of improving construction productivity and safety [3].

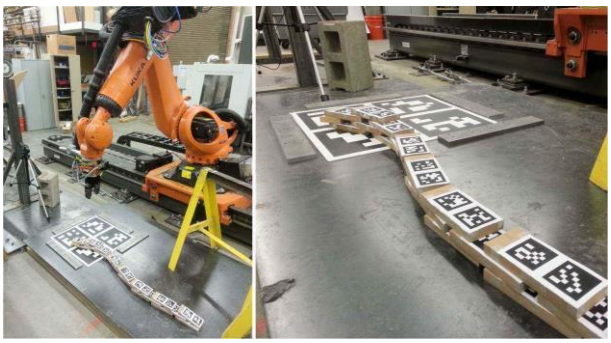

Figure 1 Curved wall assembled on-site by the designed autonomous robotic system

Compared to the tangible benefits of automation and robotics identified by the manufacturing industry, the construction industry is still exploring feasible and broadly deployable ARC applications [3]. This can be attributed to several commercial and technical challenges. From the commercial perspective, the fragmented and risk-averse nature of the construction industry leads to little investment in ARC research causing construction to lag behind other industries [4]. On the other hand, as described next, there are several technical complexities inherent in construction that have contributed to hindering the successful development and widespread use of field construction robots.

1) Unstructured Construction Environments

Automated and robotized manufacturing facilities are typically considered as structured environments, since both the machines and evolving products either stay in their predefined locations or move on predesigned and typically fixed paths. In general, such

environments do not change shape or configuration during the performance of manufacturing tasks, making the enforcement of tight tolerances possible [5]. In contrast, construction sites can typically be considered unstructured since they are constantly evolving, and dramatically changing shape and form in response to construction tasks. Building components are moved around without fixed paths or laydown/staging areas. Various physical connections are established through improvisation in response to in-situ conditions, making tight tolerances hard to maintain and enforce [6].

2) Mobility of Construction Manipulators

In manufacturing, factory robotics typically involves robotic platforms that are generally stationary (or have limited linear mobility) and partially complete products that arrive at robot workstations and precisely localize themselves in the robots' reference frames. Precision is achieved by controlling the pose of the moving (and evolving) product, and the robots themselves are programmed to manipulate the products through fixed trajectories. Thus, from a mobility and cognitive perspective, a factory robot has little responsibility and autonomy. Control is achieved by enforcing tight tolerances in moving and securing the product in the manipulator's vicinity. However, this spatial relationship is reversed in construction. A construction robot has to travel to its next workface (or be manually set up there), perceive its environment, account for the lack of tight tolerances, and then perform manipulation activities in that environment. This places a significant mobility and cognitive burden on a robot intended for construction tasks even if the task itself is repetitive.

This discussion highlights that factory-style automation on construction sites requires development of robots that are significantly more mobile and perceptive when compared to typical industrial robots. Such on-site construction robots have to be able to semantically sense and adjust to their unstructured surroundings and the resulting loose tolerances. This paper proposes a new high-accuracy 3D machine vision metrology for mobile construction robots. The developed method uses fiducial markers to rapidly establish a local high-accuracy control environment for autonomous robot manipulation on construction sites. Using this method, it is possible to rapidly convert a portion of a large unstructured environment into a high-accuracy, controllable reference frame that can allow a robot to operate autonomously (Figure 1).

The rest of the paper is organized as follows: Related work is reviewed in section 2. The authors' technical approach is discussed next in detail in section 3. The experimental results are shown in section 4. Finally, in section 5, the conclusions are drawn and the authors' future work is summarized.

2 Previous Work

The construction community has pursued ARC research for several decades. Various robotic platforms were prototyped focusing on specific construction activities (e.g., interior finishing robot [7], infrastructure inspection and maintenance robot [8], assembly robot [9], masonry robot [10]). During this research, it was realized that increasing the level of autonomy for construction robots requires high accuracy localization of the robot: from 3-5 cm indoor positional accuracy for contactless construction tasks such as spray-painting, to 2-3 mm accuracy for more precise tasks demanding direct contact between manipulator and building components [11]. This requirement has posed a significant challenge for ARC because even by using current state-of-the-art simultaneous localization and mapping (SLAM) techniques, such accuracy is hard to achieve at large scales [12]. In order to address this issue, the authors chose to use computer-vision-based pose estimation algorithms that can achieve high accuracy locally around a visual marker [13, 14].

Recently the architectural design community has also shown an increased interest in industrial robotics, with many academic programs investing in their own robotic work cells[1], leveraged as development platforms for the exploration and refinement of novel production techniques in which material behaviour is intrinsically linked to fabrication and assembly logic. As part of the general ecosystem of industrial robotics, computer vision systems have begun to play an increasingly important role in these research initiatives.

Initially the majority of architectural robotic research utilizing computer vision has revolved around its application at the *micro* scale, using vision feedback systems to make incremental adjustments to a robotic strategy based upon local variations [15, 16]. While beneficial as a means to adjust for material variation and machine error, these implementations are not robust enough for the in-situ robotics due to the complexities of construction sites.

Recently, architects have also begun to explore *macro* scale computer vision applications. At the forefront, an eight-meter-long module wall was assembled by an ABB robot along a gestural path captured by its vision system in the ECHORD project [17]. The mobile robot also used the same system to reposition itself on the construction site and make local adjustments based on topographic variation. Without using an extensive sensor suite like the one used in ECHORD, the robotic platform described in this paper successfully built similar module walls purely based on perceived information from a single camera.

[1] http://www.robotsinarchitecture.org/map-of-robots-in-architecture

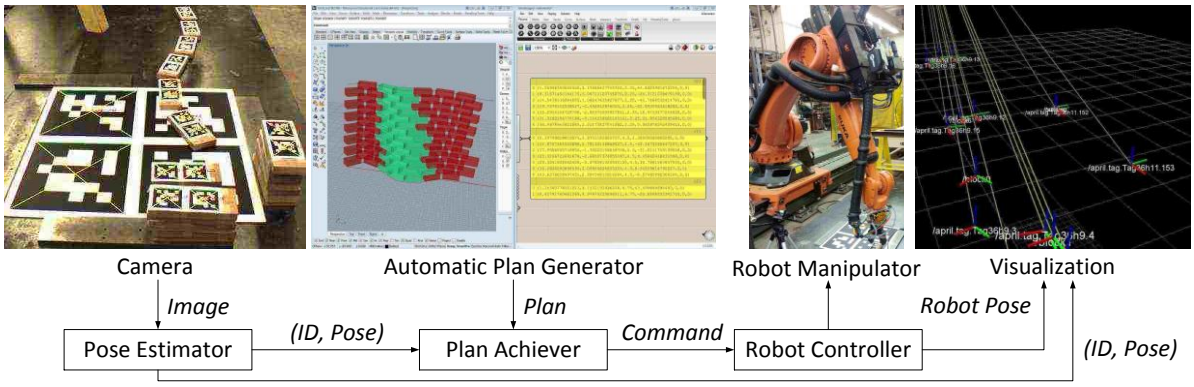

Figure 2 Technical approach

3 Technical Approach

3.1 System Overview

The developed robotic in-situ assembly system consists of 7 major components, as shown in Figure 2. The workflow of the system consists of an offline design process and an online building process. During the offline process, a designer models the intended structure in 3D, which is then analysed and validated by the automatic plan generator, outputting a building plan for the online process. The online building process uses a fixed camera mounted on the base of the robot for providing images to the pose estimator to detect staged building components and estimate their pose, and more importantly localize the robot itself in the local building reference frame. Having computed this information, the plan achiever then sequentially transforms each step from an automatically generated assembly plan into an executable command, which can be interpreted by the robot controller and subsequently executed by the robot manipulator. In addition, the visualization component also receives the information generated by the pose estimator as well as the robot's real-time pose feedback from the controller, to simultaneously represent the actual on-site assembly process into a 3D virtual environment for improved monitoring.

3.2 Calibration of Pose Estimator

Before introducing the details of other components of the system, it is important to discuss how the pose estimator is calibrated, since this is crucial to the level of accuracy that the system can achieve. This process includes two steps: intrinsic and extrinsic calibration of the camera. Intrinsic calibration involves estimating the camera's focal length, principle point's position on the image plane, and distortion parameters. On the other hand, extrinsic calibration aims to determine the relative 6-DOF pose of the camera in the robot's coordinate frame. It must be noted that these two calibrations are a one-time process, as long as the camera is fixed-focus and rigidly mounted in the robot's coordinate frame (e.g., fixed installation on the robot's base).

3.2.1 Intrinsic Parameters

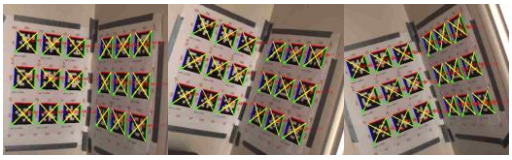

Figure 3 Intrinsic calibration of camera

Unlike the popular plane-based camera calibration method [18] implemented in OpenCV [2] and Matlab Calibration Toolbox[3], the authors chose to calibrate the camera using a 3D rig, which is similar to the classic calibration in photogrammetry. This 3D rig was made by attaching 18 Apriltags [14] on two intersecting planes forming a 90 degree angle (as shown in the top of Figure 3) so that the 3D coordinate **X** of each Apriltag's center could be readily measured.

The process of calibration was then simply taking a sequence of N images of the rig and inputting them into the author-developed camera calibration tool[4], which takes advantage of the Apriltag detection algorithm to detect the 2D image coordinate **U** of each tag center and establish correspondence with **X**. Then the initial camera intrinsic and extrinsic parameters can be obtained through Direct Linear Transform (DLT) [19] and subsequently optimized by bundle adjustment [20].

Benefitting from the high corner detection accuracy of Apriltag as well as the 3D rig, this intrinsic

[2] http://docs.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

[3] http://www.vision.caltech.edu/bouguetj/calib_doc/

[4] Available at https://code.google.com/p/cv2cg/#apriltag

calibration produces more robust and repeatable results than the alternatives mentioned above.

3.2.2 Extrinsic Parameters

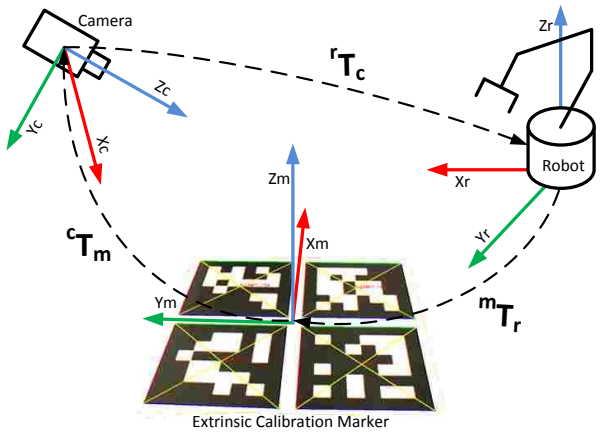

Figure 4 Extrinsic calibration

Once the camera's intrinsic parameters are calibrated, the camera's relative pose in the robot's coordinate frame, ${}^{r}\mathbf{T}_c = [{}^{r}\mathbf{R}_c, {}^{r}\mathbf{t}_c; \mathbf{0}, 1]$, can be estimated using an extrinsic calibration marker containing 4 Apriltags with known size and spacing.

As shown in Figure 4, ${}^{r}\mathbf{T}_c$ can be composed from the two other poses, ${}^{m}\mathbf{T}_r$, the robot's pose in the extrinsic calibration marker's coordinate frame, and ${}^{c}\mathbf{T}_m$, the marker's pose in the camera coordinate frame, by ${}^{r}\mathbf{T}_c = ({}^{c}\mathbf{T}_m \, {}^{m}\mathbf{T}_r)^{-1}$.

This extrinsic calibration process consists of the following steps:

1) Fix the marker in the camera's field of view;
2) Manually control the robot manipulator to pinpoint at least 4 non-collinear points on the marker and record their 3D coordinates ${}^{r}\mathbf{X}$ in the robot's coordinate frame; also measure their local 3D coordinates ${}^{m}\mathbf{X}$ in the marker's reference frame (by setting all Z coordinates to be zero);
3) Take an image from the camera and detect the 4 Apriltags' 16 corners' 2D image coordinates **U**.

With this information collected, the ${}^{m}\mathbf{T}_r$ can be estimated using the well-known rigid body registration [21] from 3D point set ${}^{r}\mathbf{X}$ to ${}^{m}\mathbf{X}$, while the ${}^{c}\mathbf{T}_m$ can be estimated by decomposing the homography between ${}^{m}\mathbf{X}$ and **U** using the previously calibrated camera intrinsic parameter **K** [18, 13].

In order to improve the extrinsic calibration's accuracy, a non-linear optimization of ${}^{c}\mathbf{T}_m$ is also performed in addition, since during the homography decomposition, a polar decomposition is performed to get a valid rotation matrix ${}^{c}\mathbf{R}_m$, which causes the result to be non-optimal. This optimization, as shown in equation (1), can be done by tuning the initial ${}^{c}\mathbf{T}_m$ to minimize the re-projection error:

$$\arg\min_{{}^{c}\mathbf{R}_m, {}^{c}\mathbf{t}_m} \sum_{j=1}^{16} \left\| \mathbf{U}_j - \mathbf{K}({}^{c}\mathbf{R}_m \, {}^{m}\mathbf{X}_j + {}^{c}\mathbf{t}_m) \right\|^2 \qquad (1)$$

3.2.3 Calibration Validation

The calibration can be validated by the following procedure:

1) Fix the extrinsic calibration marker to a *new pose* that is different from the one used in the calibration;
2) Measure, in robot frame, the 3D coordinates ${}^{r}\mathbf{X}$ of a set of M corner points on the marker (e.g., the 16 corners used previously);
3) Take an image and find out the corresponding 3D coordinates ${}^{c}\mathbf{X}$ in the camera reference frame using Apriltags;
4) Calculate the residual using equation (2):

$$\sqrt{\frac{1}{M}\sum_{j=1}^{M} \left\| {}^{r}\mathbf{X}_j - ({}^{r}\mathbf{R}_c \, {}^{c}\mathbf{X}_j + {}^{r}\mathbf{t}_c) \right\|^2} \qquad (2)$$

This residual should ideally approach zero, as smaller residual indicates better calibration accuracy.

3.3 Automatic Plan Generation

3.3.1 Algorithmic Architectural Design

Starting with the introduction of Ivan Sutherland's Sketchpad at MIT in the 1963, the architectural exploration of computation has focused on the digital environment's ability to represent an object as a system "compromised of and working with a series of interrelated systems" [22], a surprising contrast to the discrete geometric representations found in many 2D and 3D CAD applications. Initially as a domain of specialized research groups embedded in academia or commercial practices, this systems approach to digital design has become increasingly commonplace. In the 1990s many architects, unsatisfied with the capabilities presented by off-the-self software, began to develop their own software solutions through both higher-level scripting languages for CAD packages and ground-up application development. Currently, visual programming interfaces that afford designers quick

access to the potential of computation without the effort of coding syntax are available as plug-ins for popular commercial software, such as Dynamo[5] for Autodesk's Revit and Grasshopper[6] for McNeel's Rhinoceros.

The authors implemented this systems approach (and its respective tools) to automatically derive the robotic positioning data for each building block in a curved stack-unit wall from a single user-generated non-uniform rational basis spline (NURBS) surface. Working from a predetermined block size, an algorithm developed in the Grasshopper plug-in for Rhinoceros extracts latitudinal section curves from the input surface, generating a running bond pattern of the said blocks. Each block is checked for volumetric collisions with adjacent blocks, color-coding collisions in the Rhinoceros environment. Instead of applying simple heuristics to arbitrarily resolve these collisions, the color-coding can provide real-time feedback, engaging the designer to actively participate in the development of the curved stack-unit wall. Simultaneously the necessary positional information for each block is output for the generation of the building plan. Combined into a single algorithmic process, these functions "enable an explicit and bidirectional traversal of the modern division between design and making" [23], reinforcing the implications of William Mitchell's statement that "architects tend to draw what they can build, and build what they can draw" [24].

3.3.2 Building Plan Generation and Simulation

Given the final positions and orientations of all the building blocks in the design, the building plan is generated and written into a text file stored for the plan achiever to process later during the online building phase.

This building plan file contains a list of sequential instructions for the robot manipulator to build the designed structure. Each line in the file corresponds to an instruction. For example, the following plan file segment will instruct the robot manipulator to first grab a building component named "block0" directly from above (line 1-4), then lift it vertically up for 500 mm (line 5) and finally place it at its destination in another reference frame named "building" (line 6-8):

Gripper 0
Goto block0 0 0 500 0 0 0
Goto block0 -12 -10 -10 0 0 0
Gripper 1
Shift 0 0 500 0 0 0
Goto building 200.00 -300.00 500.00 -63.92 0.00 0.00
Goto building 200.00 -300.00 19.05 -63.92 0.00 0.00
Gripper 0

[5] http://autodeskvasari.com/dynamo
[6] http://www.grasshopper3d.com/

Currently, 3 types of instructions are implemented in the system:

1) **Gripper *0/1***
   Control the manipulator's gripper to open (0) or close (1);
2) **Goto *reference_frame x y z a b c***
   Control the manipulator to move to a new pose (x, y, z, a, b, c) in the reference frame, in which the (x, y, z) is the new position and (a, b, c) specifies the new orientation as three Euler angles in "ZYX" order;
3) **Shift *x y z a b c***
   Control the manipulator to incrementally move by (x, y, z, a, b, c).

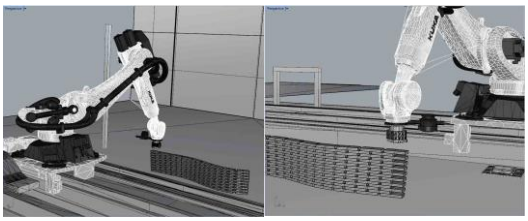

Figure 5 Building plan simulation

This building plan can also be simulated in Rhinoceros to check if there exists any self-collision between the robot manipulator and the wall during the building process, as shown in Figure 5.

3.4 Vision-based Plan Achiever

3.4.1 Rapid Setup of Building Reference Frame

As previously mentioned, the reversed spatial relationship of product and manipulator on construction sites poses a significant challenge for autonomous mobile robots. This is notably different from typical autonomous manufacturing spatial configurations, where robots' bases are either stationary or have finite mobility, and materials/components can be readily staged at fixed locations within the manipulators' static workspaces. In contract, for mobile robots to autonomously perform building tasks on unstructured construction sites, their bases require significant mobility, and consequently their manipulators' workspaces are not fixed with respect to the construction site. In order to complete building tasks at the correct locations and assemble materials into their intended poses, a robotic system must be able to establish the accurate 6-DOF transformation between the robot's base and the building reference frame at all times. As pointed out in [11], this requires the localization accuracy to be at least at centimeter level, which is far from achievable using state-of-the-art SLAM style techniques for mobile robots.

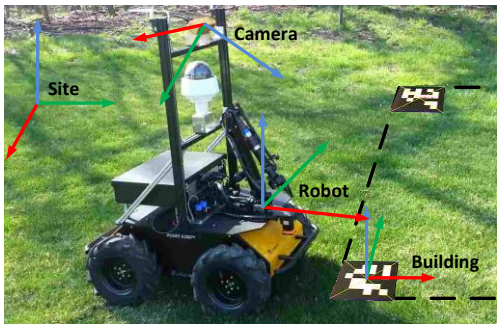

Figure 6 Different reference frames

In order to address this challenge, the authors propose a convenient and accurate solution using planar marker-based pose estimation [14, 13], as shown in Figure 6. By 1) attaching fiducial markers at appropriate locations on-site where building tasks are to be performed, 2) surveying their poses ${}^{m}\mathbf{T}_{b}$ in the building reference frame using a total station, and 3) storing these poses inside the system's database, a mobile robot can readily estimate its base's pose ${}^{b}\mathbf{T}_{r}$ inside the building reference frame using equation (3) whenever its on-board camera detects such a marker, based on previous calibration results:

$$ {}^{b}\mathbf{T}_{r} = ({}^{r}\mathbf{T}_{c}\,{}^{c}\mathbf{T}_{m}\,{}^{m}\mathbf{T}_{b})^{-1} \tag{3} $$

3.4.2 Conversion from Plans to Commands

With the information input from the pose estimator, the vision-based plan achiever starts to execute the building plan generated beforehand, according to the following procedure:

1) Read a single plan step (i.e. one line) from the building plan file;
2) Wait until all the poses needed to convert this step into a building command become available;
3) Convert this step into a command that is executable by the robot controller;
4) Send the command to the robot controller;
5) Wait for the controller to complete the command;
6) Repeat this process unless all plan steps are completed, i.e. the plan is achieved.

It must be noted that the core step of this procedure is the conversion from a plan step to an executable command. This is because the poses stored in the previously generated building plan are not completely specified in the robot's base reference frame. Recall that every pose in the "*Goto*" step is specified in a "*reference_frame*" relatively. Specifying all steps in the building plan in the robot base reference frame is not possible because: a) during the design phase the designer conceives all component locations in the building reference frame; b) the robot's base is expected to be mobile during the building phase; and c) more importantly, the building components will be arbitrarily transported and staged in the building reference frame in the vicinity of the robot manipulator's workspace. This, in fact, is one of the core differences between on-site construction automation and manufacturing automation.

In the authors' approach, this conversion is facilitated by the aforementioned rapid setup of the building reference frame using markers. As long as the pose estimator can detect and report the transformation between the on-board camera and the marker used to specify the building reference frame, the poses in the plan steps can be readily converted to the robot base frame using equations similar to (3). Similarly, by attaching markers on the building components, the robot manipulator can detect and clasp them autonomously after the corresponding plan steps are converted.

4 Experimental Results

4.1 Design and Goal

The authors implemented the designed algorithms into a robotic system using a 7-axis KUKA KR100 robotic arm, a Point Grey Firefly MV camera, and a laptop with an Intel i7 CPU, connected through the Robot Operating System (ROS)[7]. Each component in Figure 2 except for the plan generator and robot manipulator is a process corresponding to a ROS node. The camera node sends images of size 1280 pixels by 960 pixels to the pose estimator that implements the Apriltag detection algorithm in C/C++. The plan achiever was implemented in Python. The robot controller was also developed using Python to send and receive control signals via Ethernet through KUKA's native Robot Sensor Interface (RSI). Inside this controller, a 6-DOF PID control algorithm was employed to drive the robot manipulator (with a two-finger gripper) to its destination pose when executing commands from the plan achiever. The involved inverse kinematics computations are performed inside the KUKA manipulator's controlling middleware.

In the first phase of experiments, the robot was tasked with assembling a section of a curved wall, as designed in section 3.3.1. The design was shown in Figure 2. The building components used were a set of 170x70x20mm^3 medium-density fibreboard (MDF) blocks, each affixed with two different 56x56mm^2 Apriltags. The building reference frame was setup by 4 different 276x276mm^2 Apriltags. The overall goal of the experiment was to test the robot's ability to autonomously build the designed wall.

[7] http://www.ros.org/

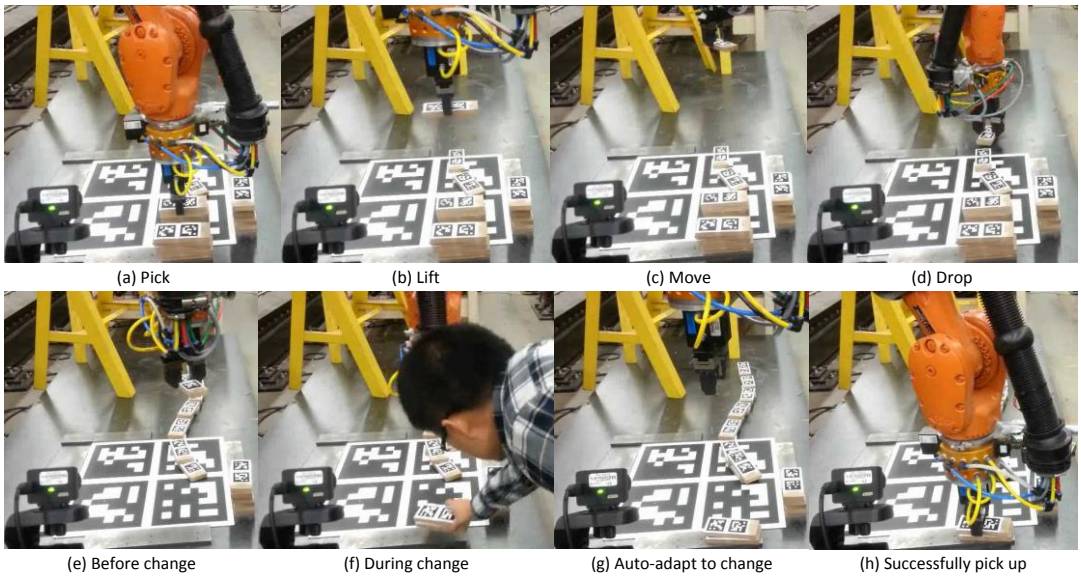

Figure 7 Autonomous assembly experiments

4.2 Results and Discussion

The system was first calibrated and validated using the methods discussed earlier. The validation residual calculated using equation (2) was found to be less than 1mm. With the accurately calibrated intrinsic and extrinsic parameters, the online building process proceeded smoothly. The building blocks were affixed with smaller markers, which decreased the building block localization accuracy to centimetre level (2cm) during the clasping process. The error was however compensated by the tolerance of the gripper.

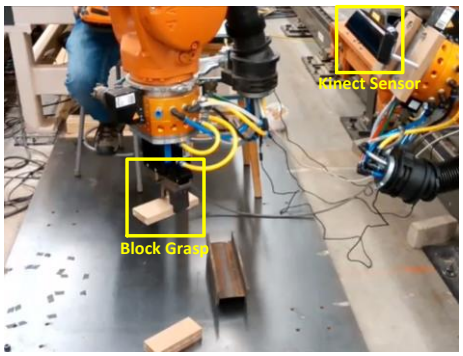

Figure 8 Block detection and grasp using Kinect

A working cycle of the autonomous building process was shown in Figure 7(a)-(d). Since the pose estimator was constantly monitoring and updating the poses of each marker, the system was naturally capable of automatically adapting to pose changes on-site. As shown in Figure 7(e)-(h), when a building block's pose changed, the robot manipulator was automatically able to pick it up at its newest location. A video recording of the experiment can be found online at the following URL: http://youtu.be/fj7AXRpj97o. A fully assembled three-layer curved wall approximately 1.5m in length was previously shown in Figure 1. Although not closely related to the core contribution of this paper, it's worth noting that the block detection and grasp can also be achieved without markers using Kinect sensor and the author's newly developed fast plane extraction algorithm [25], as shown in Figure 8 and demonstrated at the following URL: http://youtu.be/CyX4Pr_xly0.

5 Conclusion and Future Work

This paper reported algorithms and an implemented robotic system that is able to automatically generate building plans from computational architectural designs and achieve these plans autonomously on construction sites. In order to address the localization accuracy challenge, the authors proposed a computer-vision-based sub-centimetre-level metrology that enables pose estimation using planar markers. The conducted evaluation experiments used the designed robotic system to autonomously build a curved wall of MDF blocks, proving the algorithms and the system's ability to meet the accuracy requirement when building computational architecture designs.

The authors' current and planned work in this research direction is focused on continuously improving this system in aspects such as 3D perception for efficient object grasping, autonomous navigation for implementation on mobile platform, and improved and stable control algorithms.

References

[1] Rojas E. and Aramvareekul P. Is Construction Labor Productivity Really Declining? *Journal of Construction Engineering and Management*, 129(1):41-46, 2003.

[2] Bureau of Labor Statistics. Census of Fatal Occupational Injuries. Online: http://www.bls.gov/iif/oshcfoi1.htm#2009, Accessed: 17/06/2012.

[3] Balaguer C. Nowadays trends in robotics and automation in construction industry: Transition from hard to soft robotics. In *ISARC*, Jeju. Korea, 2004.

[4] Saidi K., O'Brien J. and Lytle A. Robotics in Construction. In *Springer Handbook of Robotics*, pages 1079-1099, 2008.

[5] Milberg C. and Tommelein I. Role of tolerances and process capability data in product and process design integration. In *Proceedings of Construction Research Congress*, 2003.

[6] Milberg C. and Tommelein I. Application of Tolerance Mapping in AEC Systems. In *Proceedings of Construction Research Congress*, 2005.

[7] Kahane B. and Rosenfeld Y. Real-time "Sense-and-Act" operation for construction robots. *Automation in construction*, 13(6):751-764, 2004.

[8] Kim Y.S. and Haas C.T. A model for automation of infrastructure maintenance using representational forms. *Automation in Construction*, 10(1):57-68, 2000.

[9] Gambao E., Balaguer C. and Gebhart F. Robot assembly system for computer-integrated construction. *Automation in Construction*, 9(5):479-487, 2000.

[10] Pritschow G., Dalacker M. and Kurz J. Configurable control system of a mobile robot for on-site construction of masonry. In *ISARC*, 1993.

[11] Shohet I. M. and Rosenfeld Y. Robotic mapping of building interior—precision analysis. *Automation in construction*, 7(1):1-12, 1997.

[12] Kümmerle R., Steder B., Dornhege C., Ruhnke M., Grisetti G., Stachniss C. and Kleiner A. On measuring the accuracy of SLAM algorithms. *Autonomous Robots*, 27(4):387-407, 2009.

[13] Feng C. and Kamat V.R. Plane Registration Leveraged by Global Constraints for Context-Aware AEC Applications. *Computer-Aided Civil and Infrastructure Engineering*, 28(5):325-343, 2012.

[14] Olson E. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.

[15] Dierichs K., Schwinn T. and Menges A. Robotic Pouring of Aggregate Structures. In *Robotic Fabrication in Architecture, Art, and Design*, New York, 2012.

[16] Dubor A. and Diaz G.B. Magnetic Architecture. In *Robotic Fabrication in Architecture, Art, and Design*, New York, 2012.

[17] Helm V., Ercan S., Gramazio F. and Kohler M. In-Situ Robotic Construction: Extending the Digital Fabrication Chain in Architecture. In *Proceedings of the 32nd Annual Conference of the Association for Computer Aided Design in Architecture*, San Francisco, 2012.

[18] Zhang Z. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330--1334, 2000.

[19] Hartley R. and Zisserman A. *Multiple view geometry in computer vision*. Cambridge University Press, 2000.

[20] Triggs B., McLauchlan P.F., Hartley R.I. and Fitzgibbon A.W. Bundle adjustment—a modern synthesis. In *Vision algorithms: theory and practice*, pages 298-372, 2000.

[21] Besl P.J. and McKay N.D. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239--256, 1992.

[22] Ahlquist S. and Menges A. Introduction: Computational Design Thinking. In *Computational Design Thinking*, pages 10-29, 2011.

[23] Pigram D., Maxwell I., McGee W., Hagenhofer-Daniell B. and Vasey L. Protocols, Pathways, and Production. In *Robotic Fabrication in Architecture, Art, and Design*, New York, 2012.

[24] Mitchell W.J. Roll Over Euclid: How Frank Gehry Designs and Builds. In *Frank Gehry, Architect*, page 354, 2001.

[25] Feng C., Taguchi Y. and Kamat V.R. Fast Plane Extraction in Organized Point Clouds Using Agglomerative Hierarchical Clustering. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.