Midterm Project Report:

Communicating Robot State via Real Time Projection Mapping

Henry Stone and Mckenna Cisler

May 2019

1 Abstract

Effective human-robot communication is essential to creating useful robotics applications. Since humans are visual creatures, aspects of a robot's state, intentions and understanding of its environment are frequently best communicated visually. Currently, state of the art devices for communicating such information include virtual and augmented reality head-mounted displays, but these devices are limiting; for example, they must be worn by all users and must be pre-configured to communicate with the robot. This approach replaces VR/AR headsets with a projector and depth camera attached to the robot, enabling the robot to project its state directly onto a changing environment in real time, at points specified using only its camera. We show that our approach can accurately project user-defined points onto an arbitrary environment with minimal reprojection error. Overall, this approach enables simpler and more accessible human-robot visual communication, by removing the need for individual headsets and consolidating the technology within the robot.

2 Introduction

Modern robots have sophisticated methods for acquiring information about their environment and computing their own internal state. This is necessary for robots to function, but it is also useful for robots to communicate this information and their own intentions to humans in order to facilitate shared situational awareness, collaboration, and trust. One of the most practical and informative methods of communicating this information is through visual aids. This high-bandwidth method enables humans to quickly grasp detailed information about the robot, and can allow humans to seamlessly interact with the robot using a shared spatial understanding.

Today, the three primary approaches for communicating robot state information visually are conventional displays, virtual reality displays, and augmented

reality displays. Conventional displays, which can exist on or off the robot, generally represent robot state through visualizations and simulated environments, while VR and AR displays provide stereoscopic representations of either a simulated environment incorporating robot data (as in VR) or the real world environment with overlays of robot data (as in AR). However, these approaches all have limitations. Conventional displays are constrained by their two-dimensional nature, and are not optimal for conveying 3D physical environments. VR and AR displays improve on this by presenting stereoscopic representations which overlay state directly onto either a simulated or real environment, but both require head mounted displays. Such displays must be worn by each user interacting with the robot, and additional complexity is required to ensure the displays and robot can communicate with each other. Additionally, these systems sometimes require knowledge of the user's position relative to the robot to provide accurate visual information. For many applications such displays require an unreasonable amount of preparation and specialization, limiting their accessibility and ease of use for human-robot interaction.

We propose an alternative approach to conveying robot information visually: projecting that information directly onto the environment using dynamic projection mapping. Our approach has two primary components, a RGBD camera (in our case, a Kinect V2 sensor), which obtains information about the geometry of the scene, and a projector, which enables the robot to project arbitrary images onto given locations in the scene. Our method is designed as an interface for robot developers; it simply requires a developer to specify a point in the RGBD camera space defining where (and what) they would like to project in the environment (world space), and our method will compute where the projector should project in order to color that world point. In other words, the developer will be able to project to pixels specified *within the camera image*, enabling them to easily select world points based solely on what the camera sees.

We achieve this interface by calibrating the projector-camera system to compute a direct transformation between camera and projector space. This allows us to transform individual pixels from camera to projector space, but we supplement the transformation with forward warping to transform complete images defined in camera space. Forward warping is used to handle interpolation between projected camera-space pixels and manage occlusions and other issues arising from differences in perspective between the camera and projector. OpenGL is used to implement forward warping and to post-process the warping to correct for inaccurate camera depth and unrealistic scene geometries. The depth camera is used to determine the geometry of the scene in real-time, so that differences in perspective between the camera and projector are taken into account in the transformation. This enables our software to project a complete image specified in camera space onto the environment, relying only on data from the RGBD camera.

We show that our approach, when given individual points specified in the RGBD camera's image space, is able to accurately project these points into the environment such that reprojection error (as measured from the camera's perspective) is minimized. The reprojection error represents how accurately we

can specify points in world space using points in camera space, and if this error is sufficiently low to not confuse or distract users, we can conclude that we are accurately projecting the robot's environmental understanding to users.

3 Related Work

Stereo camera calibration is a staple problem in computer vision, and has been studied in depth. Most methods for calibrating stereo cameras rely on finding correspondences between the two images [1]. In projector-camera calibration the projector cannot observe the world to generate correspondences with the camera. Instead, projectors generally employ structured light [2] which is then observed by the camera to generate correspondences. Certain structured light patterns used in projection mapping allow the construction of a dense correspondence between camera and projector space. Given a static scene, methods which compute these dense correspondences can project on desired locations in camera space, or compute a depth map of an observed scene [2]. However, while these methods produce high quality projection maps, changes in a scene require a new structured light scan of the environment, making real time projection mapping infeasible.

Other techniques make use of an initial camera-projector calibration alongside printed infrared patterns to construct sparse correspondences [3]. Using high frame rate projectors and cameras, such methods are able to project onto moving surfaces at over 1000 frames per second enabling a seamless visual experience. However, the requirement of printed patterns limits this method to projecting onto prepared surfaces. While other high speed projection mapping methods exist which use camera tracking of an object instead of a depth map [4], this system requires expensive non-commodity hardware, and is insufficiently mobile for small to medium robotics applications.

Most similar to our work, a recent paper [5], also incorporates a Kinect v2 for real-time projection mapping on a moving scene, performing many similar calibration steps. Their work, however, is not robotics facing, and thus does not explore ways to represent internal robotic state. Other works focus on calibration and projection mapping using the Kinect itself as a projector-camera system, and how this can be utilized to improve upon the depth estimation capabilities of the Kinect [6].

Within the domain of robotics, various works explore the value of augmented reality and virtual reality headsets for showing robot intent [7] and interactively communicating with robots [8]. Meanwhile Omidshafiei et al. [9] uses projection mapping to represent a robot's environment and understanding of its location. However, while projection mapping is used to represent the robot's internal state, the approach is limited to only projecting onto a static planar surface below the robot, and the projectors themselves are not mounted on the robot.

The main contributions of our approach are as follows

- Using Kinect data to calibrate a projector/Kinect system.

- Using Kinect depth to allow real time projection mapping onto changing scenes.
- Projecting information regarding a robot's state onto its environment.
- Mounting the projection mapping system onto a mobile robot to allow operation in a non-prepared environment.

4 Technical Approach

The primary goal of this project was to properly calibrate and project to points in camera space utilizing the Kinect's depth data. Formally, given a 2 dimensional camera point, $\{x_c, y_c\}$, and a corresponding depth, $\{d_c\}$, we would like to be able to produce a projection point, $\{x_p, y_p\}$, which corresponds to the same world point as $\{x_c, y_c, d_c\}$.

4.1 Model of a Camera

Following traditional computer vision literature, we consider a mathematical model of a camera (or a projector) composed of three elements. $\mathbf{E} \in \mathbb{R}^{4\times4}$, the extrinsic matrix, which represents a rotation and translation of world space coordinates so that the focal point of the camera lies at $(0, 0, 0)$ and the camera points in the positive Z direction with the top of the camera pointing in the positive Y direction. $\mathbf{I} \in \mathbb{R}^{4\times4}$, the intrinsic matrix, maps coordinates from this space into homogeneous image space. The intrinsic matrix represents information such as the center of the image and the cameras focal length. These two matrices are all that is necessary to represent an ideal point camera, mapping world coordinates $\vec{X} = (X, Y, Z)$ to image coordinates $\{x_c, y_c\}$ and depth $\{d_c\}$ as follows.

$$\begin{bmatrix} x_c \cdot d_c \\ y_c \cdot d_c \\ d_c \\ 1 \end{bmatrix} = \mathbf{IE} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

From this we can recover x_c and y_c by diving through by the third element of the vector. $\mathbf{I}$ and $\mathbf{E}$ both have $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$ as their bottom row, guaranteeing that the bottom index stays 1 under multiplication. If we know the depth d_c in addition to the image coordinates, we can reconstruct the vector $\begin{bmatrix} x_c \cdot d_c \\ y_c \cdot d_c \\ d_c \\ 1 \end{bmatrix}$. Then we can find the world coordinates corresponding to that point by inverting the intrinsic and extrinsic matrices.

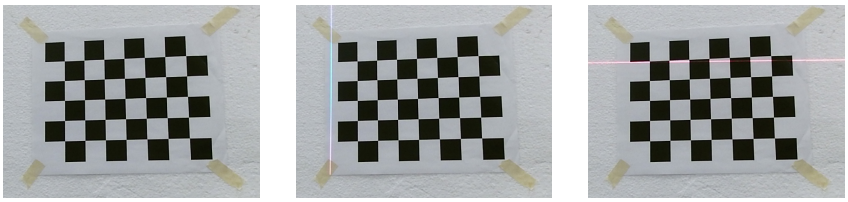
$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{E}^{-1}\mathbf{I}^{-1} \begin{bmatrix} u \cdot d \\ v \cdot d \\ d \\ 1 \end{bmatrix}$$

While this effectively models an ideal camera, in practice additional types of distortion are caused by a cameras lenses. Thus we also model distortion coefficients ρ such that the real coordinates in image space are described by a function of x_c, y_c, and ρ. For more information on specifics see the OpenCV calibration page.

4.2 Full System Calibration

Our initial goal was to fully model the both the camera and the projector, allowing transformation of coordinates to and from a mutual world space as an intermediary. Using [10] as implemented by OpenCV, we can solve for **E**, **I**, and ρ using a set of correspondences between world space and camera space (or projector space).

In order to do this, we used a camera calibration grid as our world space, and find various world space points in camera-space to perform camera calibration (figure 1a). Once we have calibrated the camera, we project lines onto the calibration grid (figure 1bc). Finding the intersections of these lines, and using planarity constraints, we can find the world points which correspond to a specific pixel in the projector (Thus enabling projector calibration). From the full calibration of the camera and the projector, we were able to covert between the two domains.



(a) Calibration grid

(b) Calibration grid with vertical projected line

(c) Calibration grid with horizontal projected line

Figure 1: Calibration Images

However, in estimating the camera matrices and distortion coefficients for both the camera and the projector, several sources of error were included that made the calibration sub-par. In particular, given the limited number of images we could process using the line intersection method, distortion coefficients tended to be higher that the expected true coefficients for the low-distortion cameras (which visually appeared to be 0). Thus we eventually discarded this approach in favor of a direct calibration.

4.3 Direct Camera Projector Calibration

If we assume that the distortion coefficients of both the camera and the projector are $\vec{0}$, the equations for projection into both the camera and the projector are simply:

$$\begin{bmatrix} x_c \cdot d_c \\ y_c \cdot d_c \\ d_c \\ 1 \end{bmatrix} = I_c E_c \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x_p \cdot d_p \\ y_p \cdot d_p \\ d_p \\ 1 \end{bmatrix} = I_p E_p \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

By using both of these equations together and completely ignoring the intermediate values we can represent the entire transformation using a single camera matrix M.

$$\begin{bmatrix} x_p \cdot d_p \\ y_p \cdot d_p \\ d_p \\ 1 \end{bmatrix} = I_p E_p E_c^{-1} I_c^{-1} \begin{bmatrix} x_c \cdot d_c \\ y_c \cdot d_c \\ d_c \\ 1 \end{bmatrix} = M \begin{bmatrix} x_c \cdot d_c \\ y_c \cdot d_c \\ d_c \\ 1 \end{bmatrix}$$

Here M is a a 4 by 4 matrix, but since we know the bottom row is $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$, we can functionally consider $M \in \mathbb{R}^{3\times 4}$. This matrix represents the entire transformation between camera-space and projector-space. In order to solve M, through minimization of reprojection error, we must collect a set of correspondences directly between camera-space and projector-space.

4.3.1 Collection of Correspondences

In order to collect correspondences, we project three different colored circles (red, green, and blue) to random (non-adjacent) coordinates in the projector. We collect images from the Kinect both before and after the circles are projected, using the difference in the images to remove objects that remain static in the scene. We blur this difference image with a Gaussian blur with $\sigma = 8.5px$ so that brighter circles in the image will have peak brightness at their center. Finally, taking the maximum relative brightness of each channel (e.g. $R - \frac{G}{2} - \frac{B}{2}$), we localize the center of the projected circle in camera space. Using this location, and the Kinect's depth data, we can also find the depth of the corresponding point.

Given a set of potential corresponding camera-space points $\{x_c, y_c, d_c\}$ and projector-space points $\{x_p, y_p\}$, we first validate them by hand to eliminate clearly extraneous pairs. The remaining correspondences are used to solve for the optimal calibration, M.

4.3.2 Solving for the calibration matrix, M.

Given a set of correspondences $\{x_c, y_c, d_c, x_p, y_p\}$ we find the calibration matrix, M, as follows using a variation of the 8-point algorithm. While we do not know d_p explicitly, we know that it must be equal to

$$d_p = M_{3,1} x_c d_c + M_{3,2} y_c d_c + M_{3,3} d_c + M_{3,4}$$

. Using this we know the value of x_p should be

$$\frac{M_{1,1}x_c d_c + M_{1,2}y_c d_c + M_{1,3}d_c + M_{1,4}}{M_{3,1}x_c d_c + M_{3,2}y_c d_c + M_{3,3}d_c + M_{3,4}} = \frac{x_p \cdot d_p}{d_p} = x_p$$

We simplify this by multiplying through by the denominator and moving all terms to one side of the equation. The result is an inequality in the following form for each correspondence.

$$M_{1,1}x_c d_c + M_{1,2}y_c d_c + M_{1,3}d_c + M_{1,4} - M_{3,1}x_c d_c x_p - M_{3,2}y_c d_c x_p - M_{3,3}d_c x_p - M_{3,4}x_p = 0$$

Similarly, we can construct an inequality using y_p for each correspondence. Since we know the values of $x_c, y_c, d_c, x_p,$ and y_p, each of these inequalities is simply a linear constraint on the values of M. We then find a matrix M, which satisfies these soft linear constraints as well as possible while having $\|M\|_2 = 1$ in order to avoid the trivial solution $M = 0^{3\times 4}$.

4.4 Forward Warping with OpenGL

While being able to map points from $\{x_c, y_c, d_c\}$ to $\{x_p, y_p\}$ fully captures the calibration between the Kinect and the projector, there are several significant issues to address before we can transform images in Kinect-space to images in projector-space. First, for each pixel in Kinect-space we need to apply M in order to convert the pixel to projector space. Since there are over 2 million pixels in the Kinects field of view, the required number of operations, if CPU-bound, would severely limit the frame rate of the projection. Second, if adjacent Kinect pixels are more than 1 pixel apart in projector-space, then we need to interpolate between them so that intermediate pixels are not black.

In order to solve these issues, we convert the Kinect image into a mesh, allowing much of our computational pipeline to be applied using graphics techniques on the GPU. First, we use a custom C library linked with Python in order to move the $\{x_c, y_c, d_c, R, G, B\}$ values for each pixel of the Kinect's image to the GPU. The GPU uses a preconstructed mesh of triangles to connect these vertices into a sheet.

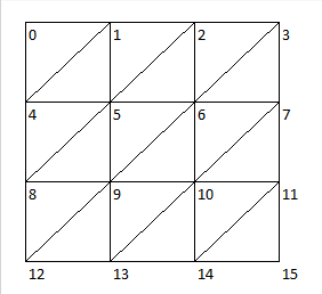


Figure 2: Grid constructed with $\{x_c, y_c, d_c, R, B, G\}$ information placed at each vertex.

Once the Kinect image data is converted into a mesh, it undergoes several stages of processing.

First, for each vertex in the mesh, the $\{x_c, y_c, d_c\}$ information is converted to $\{x_p, y_p\}$ using the matrix M. Operations of this type are integral to graphics, and thus are highly optimized on the GPU, allowing the throughput necessary to acheive real time performance.

Second, while we wish to interpolate between nearby points in projector-space, if two adjacent points in Kinect-space are very far apart in projector-space, it is likely due to an occlusion (region visible to the projector which is not visible to the Kinect). In order to avoid projecting into the region not visible to the Kinect, a geometry shader is used to remove those triangles which are very significantly streched.

Finally, a fragment shader uses the triangle connectivity to interpolate between the RGB values at each vertex in order to fill in intermediate regions in the projector image.

All the operations we perform on the GPU mirror operations which would be performed in normal real time graphics applications. Thus, GPUs are able to perform the pipeline very fast compared to what would be expected on the CPU. The whole pipeline runs at approximately 25 FPS, but could be further optimized as a significant amount of the computational cost comes from type conversions and passing the data between Python and C.

4.5 Robot Operating System (ROS) Interface

We implemented our system as a ROS package, with the projector serving as a ROS node. The projector interface was simple; the node subscribed to two topics, a `depth` topic directly from the RGBD camera, and an `image` topic which a client node would publish to in order to project. This interface is useful because the image sent from the client application to the `image` topic is in camera space, such that the client application could directly use the camera's input to generate the desired projection.

A normal use case would then involve three ROS nodes; a RGBD camera (Kinect2 in our case), projector, and a client application node. The RGBD camera provides the latest depth directly to the projector's `depth` topic, and the client application receives the RGB data from the camera to generate its own image to project, which is then sent to the projector's `image` topic.

5 Evaluation

5.1 Projection Accuracy

The purpose of calibrating a projector-camera system is to generate accurate correspondences between the two in real time. This would mean that for an arbitrary location in camera space, it is possible to find the corresponding point in projector space such that when projected it appears at that coordinate in the

camera. To evaluate this, we took 39 human-annotated ground truth projector-camera correspondences. The locations in camera space are projected using our method and compared to the ground truth projection locations which generated these points.

We compare our projection method to one which uses simple image stretching without incorporating Kinect depth.

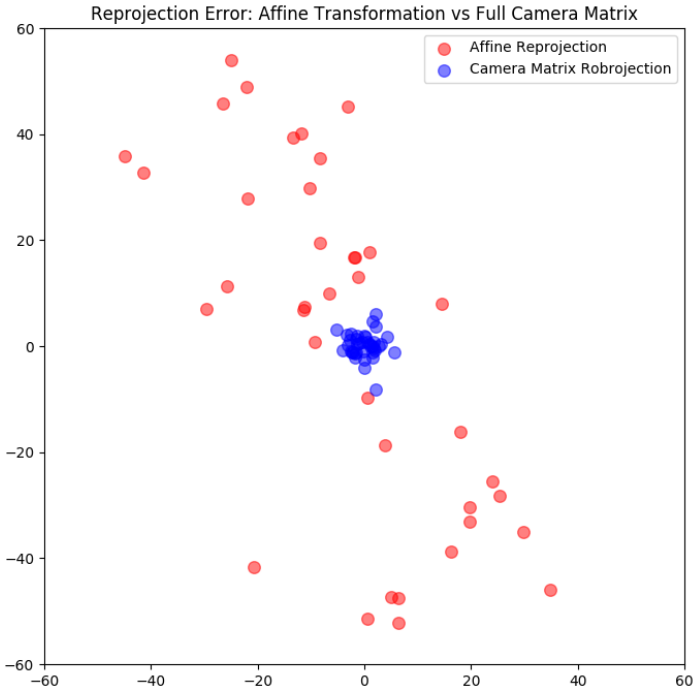


Figure 3: A comparison of projector-space reprojection error for fitting a full camera matrix (including depth) vs. an affine transform (without depth) to out calibration data. Camera space reprojection error, which is much harder to measure, would be proportional to this error although diminished by a small factor of 1.5-3, based on the resolutions of the two systems.

There are several cases where we would expect the projector/camera space transform to be affine. If the projector and camera are in the same location, or if the space being projected onto was a static plane, then the affine transformation matrix would be sufficient to capture the difference between the two spaces. When we compare such a transform fitted to our data with a full camera matrix transformation which takes into account the Kinect depth, we see that depth

information in imperative for a proper calibration even with a baseline between the Kinect and the projector as small as 15cm.

Qualitatively, we can see that the calibration performs well by comparing the pattern being projected with the pattern observed by the camera.

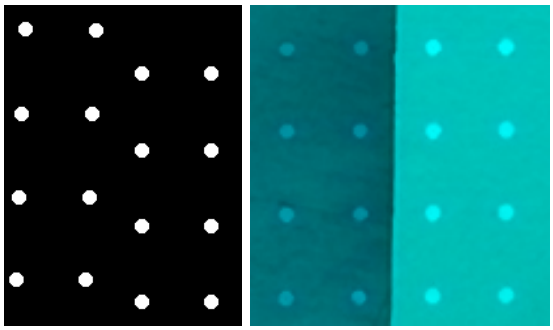


Figure 4: In an attempt to project a uniform grid in camera space (right), we see that the projection (left) must distort itself across depth boundaries.

Thus it is clear both quantitatively and qualitatively that the calibration system is capable of adequately projecting points from camera space to projector space.

5.2 Use Case Demonstrations

We also produced three demonstrations of potential use cases of our implementation of projection mapping. These were intended to qualitatively demonstrate the usefulness of this method in various robotics contexts.

5.2.1 Drawing

The drawing demonstration is a simple interface that allowed a user to draw on an image from the RGBD camera (using their mouse), and then have what they drew projected on the environment at the same location they drew on in the image.

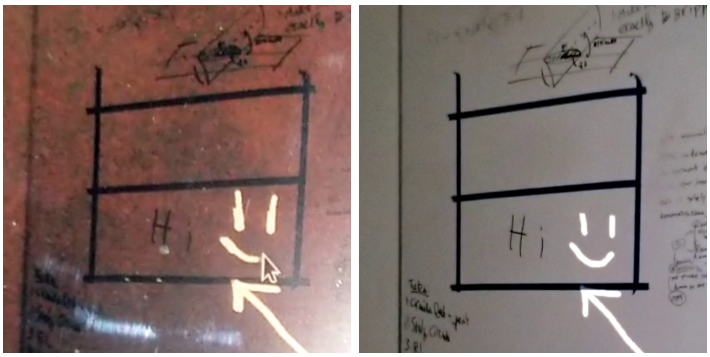


Figure 5: A demonstration of the drawing interface. The user draws directly on an image from the camera using their mouse (left), and the drawing is projected on the real world as it was drawn (right). A video demonstration can be found here. Note that the image drawn here was slightly outside of the area the projector could illuminate, and the top of the image was cut off.

This demonstration provided qualitative validation of the reprojection accuracy of our method; under visible scrutiny the drawn image appeared in the same location in the environment as was specified using the camera. In other words, the drawing on the interface is nearly perfectly overlaying the real-world projection in the left image above.

This demonstration also suggests several possible robotics applications of our software. These include:

- Enabling interaction through telepresence robots, by allowing the remote user to project on the environment.
- Novel human-robot interaction using automated drawing as a form of robot communication.
- User interfaces for controlling robots utilizing nearby surfaces.

5.2.2 Object Tracking

The object tracking demonstration shows the addition of tracking algorithms to project onto a consistent point on an *object*, rather than just a consistent point in a static scene.

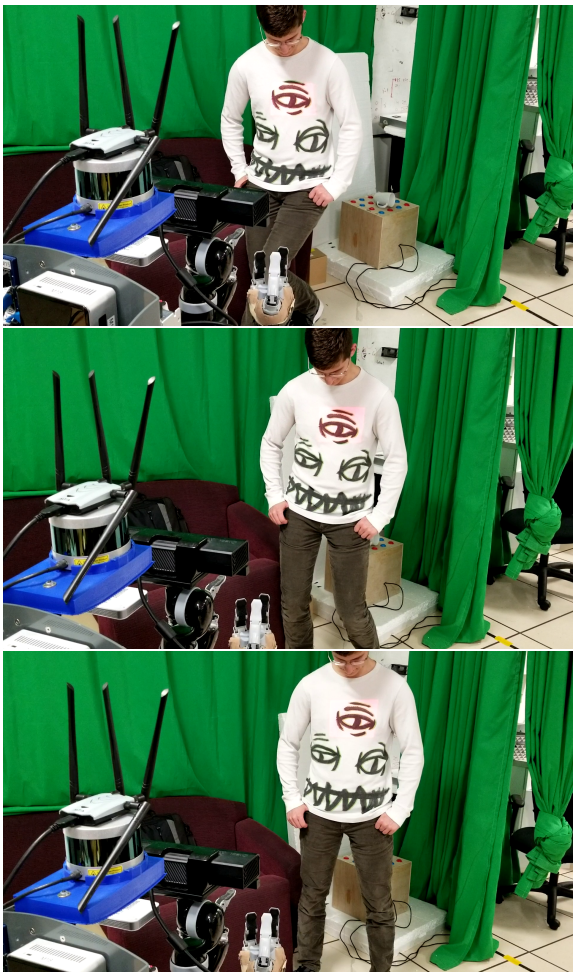


Figure 6: A demonstration of combined projection and object tracking. An object in the camera's view is selected to track (in this case, the uppermost eye on the shirt), and a red square is continually projected at the tracked object's camera coordinates as the object moves. A video demonstration can be found here.

This demonstration shows the usefulness of object tracking for enabling consistent projection onto the environment despite robot movement or changes in the environment. This example also demonstrates the relatively low latency of this system.

There are several possible applications of the combination of projection mapping and object tracking, including:

- Object illumination for human-robot communication.
- Locking human- or robot- generated projections to particular objects (for

example, as an extension to the drawing demo above).

5.2.3 MaskRCNN

The MaskRCNN demonstration shows how our system can convey high level robot understanding of its environment. MaskRCNN arrives at its labeling of the world through a complex and error-prone process. Since the system is so complex, understanding (and debugging) a robots interpretation of a scene without access to the robots internal state is very difficult. Our MaskRCNN demo shows the capability of our system to express this complex internal state in a salient and human-understandable manner.

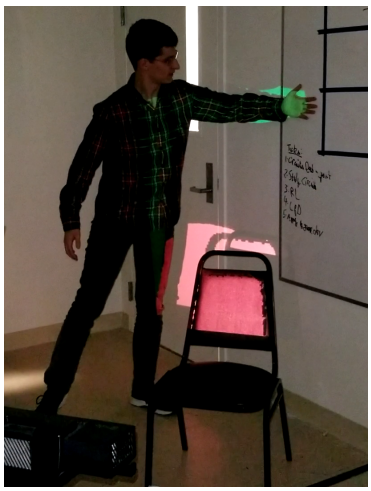


Figure 7: The MaskRCNN demonstration correctly categorizing human and chair objects. A video demonstration can be found here.

Some examples where expressing MaskRCNN information with projection mapping could be useful:

- Settings where it is important to know if a robot recognizes you as a human such as working with industrial robots.
- Debugging human-robot interaction technologies where the robot's labels of the environment are important.

6 Conclusion

This paper has presented a novel approach to conveying visual information from a robot using dynamic projection mapping. We have presented a system comprised of a depth camera and projector, which uses an initial co-calibration to produce a single transformation matrix that is used along with forward warping to project to world coordinates specified in camera-space. We have shown

the advantages of this system system over conventional VR and AR approaches to communicating robot state, and demonstrated that our method enables accurate projection of visual data onto a robot's environment due to acceptable visual error and minimal reprojection error relative to a baseline method. We have also provided demonstrations of our method that are useful for several robot applications.

Given the ability to project directly from a camera-space image (as opposed to a set of points), there are several applications that immediately present themselves. For example, a robot could use a combination of MaskRCNN and other robot processing to constantly project its classification of and intentions with objects surrounding it. This could increase the safety of human-robot interaction in collaborative work spaces, by allowing all humans in that space to understand what a robot's intentions are. Another application to consider is, given a path indicated in the Kinect's view, draw a series of footsteps on the ground, indicating the intended path of the robot. These applications both make use of the calibration to bridge between camera-space and projector-space in order to assist in expressing a robot's understanding of its state and its actions.

Our technique is limited by several factors. Notably, we only calibrate between the projector and the Kinect to which it is connected. A broader calibration scheme would also include calibration between the Kinect and other cameras on the robot, allowing the robot to project from the point of view of other cameras as well. Another extension of our technique would be to track objects in world space automatically, built off of the robot's global localization. Our object tracking demonstration showed tracking of objects in the Kinect's view, but if the Kinect were to lose sight of the desired object the track may not be recovered. One could instead correspond objects to their world coordinates, and project onto those objects whenever they were visible to the projector. One could easily imagine an application such as telepresence where you might want to project notes onto a whiteboard and have the notes reappear even if the telepresence robot were to move away and return.

Ultimately, dynamic projection mapping enables a myriad of applications for seamlessly communicating a robot's state, environmental understanding, and intentions. There is also significant opportunity to build additional features on top of projection mapping to present higher-level interfaces for roboticists, making it easier to communicate complex internal representations easily and intuitively to robot users with minimal programmer effort.

References

[1] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):965–980, 1992.

[2] Daniel Moreno and Gabriel Taubin. Simple, accurate, and robust projector-camera calibration. In *2012 Second International Conference on 3D Imag-*

ing, Modeling, Processing, Visualization & Transmission, pages 464–471. IEEE, 2012.

[3] Gaku Narita, Yoshihiro Watanabe, and Masatoshi Ishikawa. Dynamic projection mapping onto deforming non-rigid surface using deformable dot cluster marker. *IEEE transactions on visualization and computer graphics*, 23(3):1235–1248, 2017.

[4] Kohei Okumura, Hiromasa Oku, and Masatoshi Ishikawa. Acitve projection ar using high-speed optical axis control and appearance estimation algorithm. *2013 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2013.

[5] Yundong Guo, Shu-Chuan Chu, Zhenyu Liu, Chan Qiu, Hao Luo, and Jianrong Tan. A real-time interactive system of surface reconstruction and dynamic projection mapping with rgb-depth sensor and projector. *International Journal of Distributed Sensor Networks*, 14(7):1550147718790853, 2018.

[6] Thiago Motta, Manuel Eduardo Fernández, Luciano Soares, and Alberto Raposo. Projection mapping for a kinect-projector system. *Proceedings - 2014 16th Symposium on Virtual and Augmented Reality, SVR 2014*, pages 200–209, 09 2014.

[7] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. Communicating robot arm motion intent through mixed reality head-mounted displays. *arXiv preprint arXiv:1708.03655*, 2017.

[8] Hangxin Liu, Yaofang Zhang, Wenwen Si, Xu Xie, Yixin Zhu, and Song-Chun Zhu. Interactive robot knowledge patching using augmented reality. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1947–1954. IEEE, 2018.

[9] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Yu Fan Chen, Nazim Kemal Ure, Shih-Yuan Liu, Brett T Lopez, Rajeev Surati, Jonathan P How, and John Vian. Measurable augmented reality for prototyping cyberphysical systems: A robotics platform to aid the hardware prototyping and performance testing of algorithms. *IEEE Control Systems Magazine*, 36(6):65–87, 2016.

[10] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.