# ÜBERBEAMER

Thesis Project
Eric Mika

Masters Degree Candidate NYU / ITP
eric@ericmika.com

**Keywords**
spatial mapping, projection mapping, Kinect, depth map, point cloud, SLAM, projection, open source, augmented reality, light, handheld, flashlight, camera, movement, architecture, space, odometry, openFrameworks, self contained, beamer, überbeamer, peephole, pointing, window, spatial, 3D, computer vision, intrinsic projection

## Abstract

The überbeamer is a handheld projection system designed to function as a bridge between real-world and virtual surfaces. The projection selectively reveals layers of digital content on walls, floors, ceilings, or any other flat surface in a spatially consistent way — this means that content is fixed to the space such that an object projected in one corner of a room will remain in that corner of a room whether or not the projection is pointed there at any given moment. Beyond revealing  and navigating content, the system also acts as a cursor or brush through which new projected content can be added to the space, or existing elements can be manipulated. The überbeamer is essentially a read / write flashlight for any space.

The system does not depend on QR codes or other external markers to keep track of the projection's position, nor does it require a pre-recorded three dimensional model of the projection space. Rather, by pairing the projector with a depth camera (a Kinect, in this case), the überbeamer builds a model of its environment in real time. Software keeps track of the user's position relative to the geometry of the room, and calculates which content should be revealed and how that content should be mapped to appear in a spatially consistent way. Since the projection is handheld, walking and panning through a physical space with the device becomes a means of exploration and interaction.

Software development, hardware refinements, and experimentation with interaction models are all ongoing. Much remains to be discovered (and implemented). In the short term, the überbeamer exists as a convenient prototype for experimentation with what I am calling *intrinsic projection* for lack of a term of art. The long-term goal of the überbeamer project is to build an open-source hardware and software platform to allow others to realize their ideas by focusing on the creative rather than technical challenges associated with this novel approach to augmented reality.

**Project Description**

The überbeamer is a means of making any surface writable. It takes the basic premise of augmented reality — the mediation of physical space via an overlay digital information — and expands upon it by introducing a new, universal, and direct means of interacting with a space. Instead of using an iPhone as a window to an augmented world, the überbeamer flattens the information layer and the surface layer by using a projector (a.k.a. a beamer) to draw content directly onto any surface. By building and storing a three dimensional model of an environment on the fly, the überbeamer knows where you are (relative to where you've been) and can project its overlay in a spatially consistent way. Since you're holding the projector, it becomes an intuitive navigation interface — just point it where you want to see, or point it where you want to write.

The system is realized through a chain of commodity hardware, consolidated into a single handheld device via a custom-designed enclosure. (For now, handheld means "sling it over your shoulder", not "put it in your pocket". See figure 1 for a sense of scale.) Custom open-source software ties it all together. A Microsoft Kinect is the keystone that makes the überbeamer possible. Previously, hand-pointed dynamic projection systems depended on external systems to keep track of your location and heading in the projection space (Cao 2006). Thanks to the Kinect, a primitive sense of space and orientation is available in a consumer-grade piece of hardware. Spatial reconstruction is possible through a computer vision  algorithm called simultaneous localization and mapping — or SLAM. SLAM can be implemented through various combinations of algorithms, but in the case of the Kinect's point clouds iterative closest point (ICP) is used to construct a spatial model. ICP compares and aligns several frames of point cloud data from the Kinect's depth camera, building a perpetually expanding cloud model

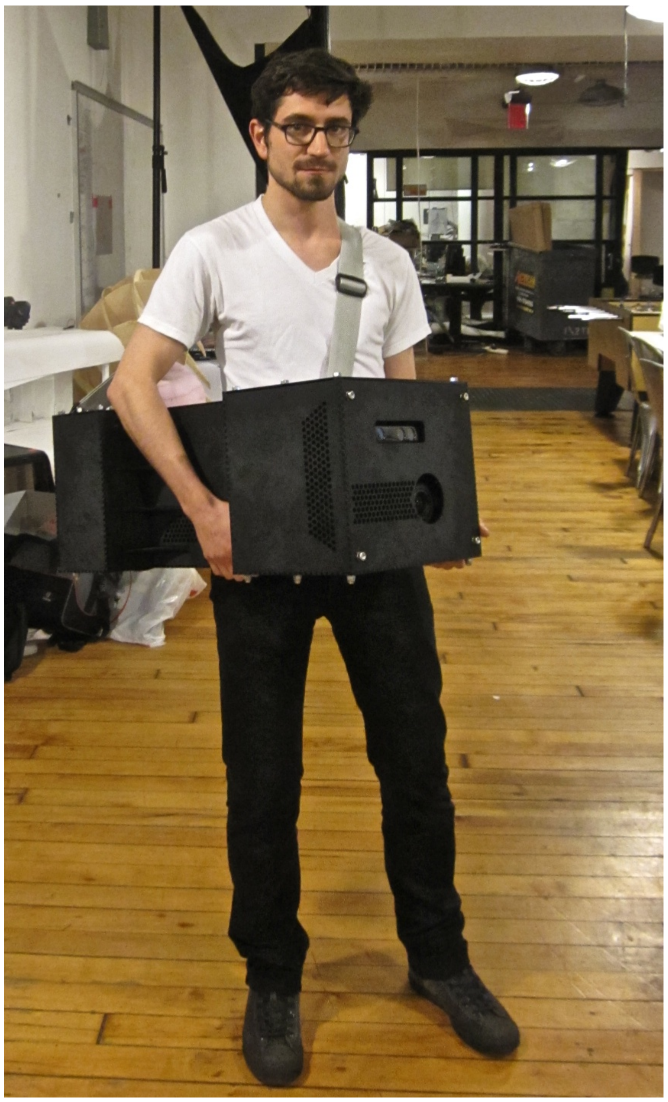


Figure 1: The überbeamer in hand.

of the environment — so long as the Kinect keeps moving through the space. In addition to mapping space, SLAM tells you where you are within the model and where the Kinect is pointed. This algorithm is nothing new, it's been used in robotics applications in combination with LIDAR (Light Detection And Ranging) systems since the mid 1980s. The Kinect just brings the integration complexity and costs down by an order of magnitude, and opens up a range of new interaction possibilities for developers and artists without the resources of a formal research setting.

The überbeamer builds on the Kinect's spatial mapping capabilities by adding a projector to the equation. Careful alignment and stabilization of a Kinect and a projector enable the core functionality of intrinsic protection — simultaneous re-projection into real space using position information derived from the SLAM algorithm. The Kinect's color camera — which is situated alongside the depth sensor — also holds potential. Ingesting and re-projecting chunks of the environment opens up another layer of interaction possibilities worth exploring.

## Project vs. Platform

There are many disciplines and industries which might find applications for intrinsic projection. Museums, TV and film, software interface design, team collaboration tools, architecture, construction, and education all come to mind immediately. The überbeamer integrates a blend of recent and relatively familiar technologies — I wouldn't quite call it a fundamentally new medium — but the interaction model is unique enough that we don't have a library of "known knowns" or best practices from which to draw.

Given this vague but sincere sense of potential usefulness, the initial goal of the überbeamer project is to build a hardware and software platform that's well suited to rapid prototyping and software iteration. There is not yet a killer app for intrinsic projection, but my hope is that the überbeamer's open-source hardware and software design will improve the odds of finding the killer app and building a vocabulary of interface elements and approaches.

Rather than build a single-purpose system, the überbeamer was designed with as a generic reference implementation capable of running a range of possible applications. At a fundamental level, I'm building the platform to satisfy my own curiosity about intrinsic projection and to support a number of application ideas I have in mind. At a more general level, I'm building the first prototype with ITP students and their ilk in mind. (Specifically, creative technologists who work in an empirical and iterative way.) In the future, specific applications or use cases — the natures of which are presently unknown — may be better suited to more task-specific hardware designs. For example, the enclosure design could be modified for lighter weight, additional buttons, sound capabilities, etc.

## Hardware Design

The entire hardware setup is as follows. Keep in mind that everything is wrapped up in a hand-held enclosure: A compact projector mounted in alignment with a Kinect. Both are plugged into a laptop (mounted in the same enclosure, see figure 2) running a version of the SLAM algorithm optimized for the überbeamer's use case, which involves using the aforementioned ICP algorithm for point cloud alignment instead of the RGB-space feature-tracking techniques employed by most SLAM implementations. (Software design is discussed at length in the next section.) A few peripheral elements keep everything running. An Arduino connected to the laptop coordinates a small servo motor to auto focus the projector based on surface distance information collected from the Kinect. (Integrated autofocus is currently not available for projectors compact enough to be useful for this application.) A simple two button interface is also connected to the Arduino. Beyond the act of aiming and sweeping the projector across the room, these two buttons are the only point of input. Their exact use is up to the application developer, but the choice of two buttons is inspired by the universal left-click (select) and right-click (context / mode switching) paradigm. In the prototype unit, a commercially available laptop docking system called a Henge Dock (see figure 3) simplifies connection of a laptop to the rest of the hardware (and avoids forcing you to permanently entomb a laptop in the enclosure, or having to fuss with a bunch of wires every time you want to use the überbeamer). Combining the Kinect, laptop, and projector in close quarters generates a significant amount of heat. To address this, a cooling fan mounted at the front of the

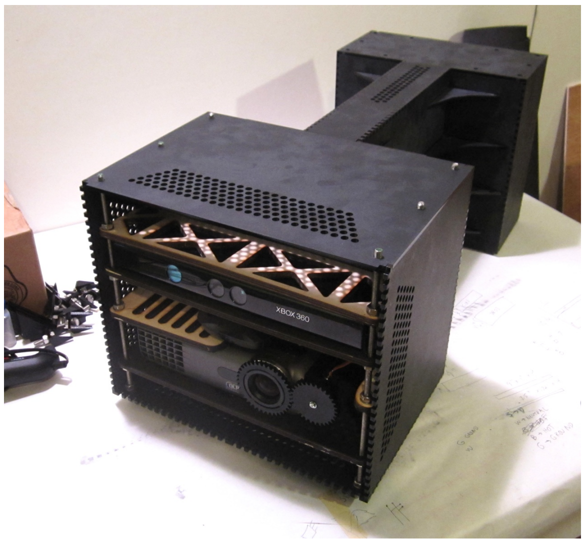


Figure 2: The projector and Kinect visible after removal of the front panel.

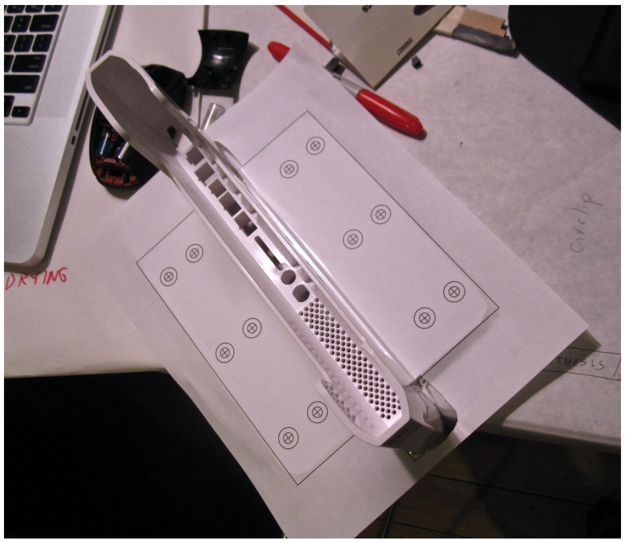


Figure 3: The henge dock, which allows for quick connection and disconnection between the laptop and the überbeamer.

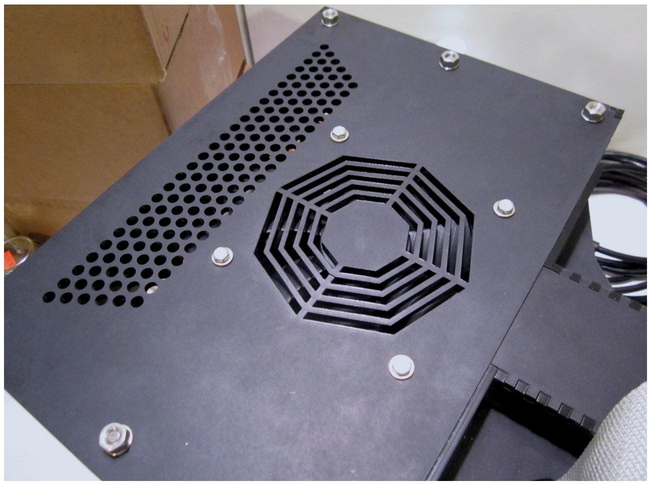


Figure 4: A cooling fan installed on the top of the projector pod.

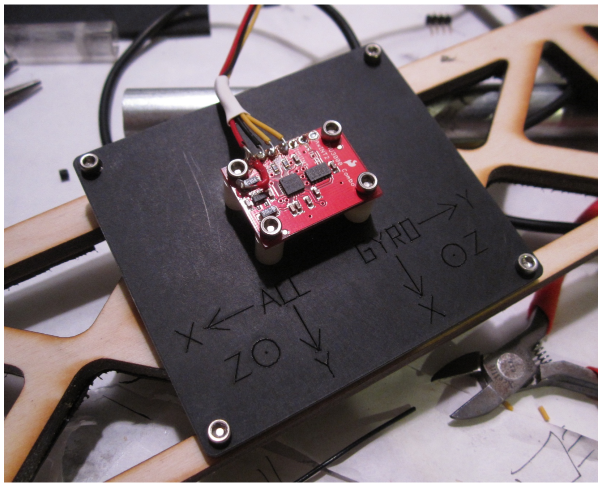


Figure 5: IMU (Gyroscope and accelerometer to assist the ICP algorithm.)

enclosure and the projector is run in "high altitude mode", which increases the projector's fan speed at the expense of a bit more noise. (see figure 4)

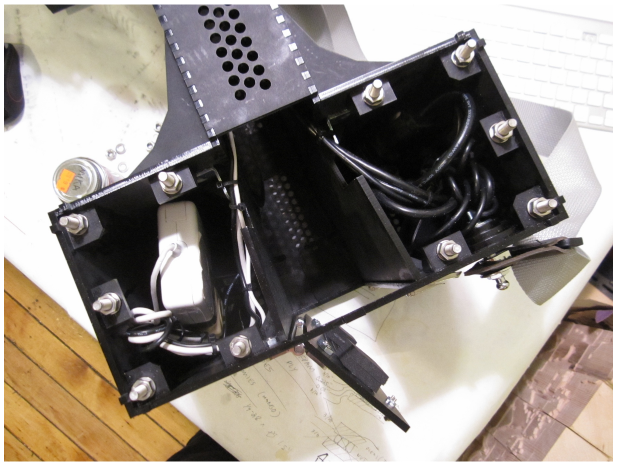
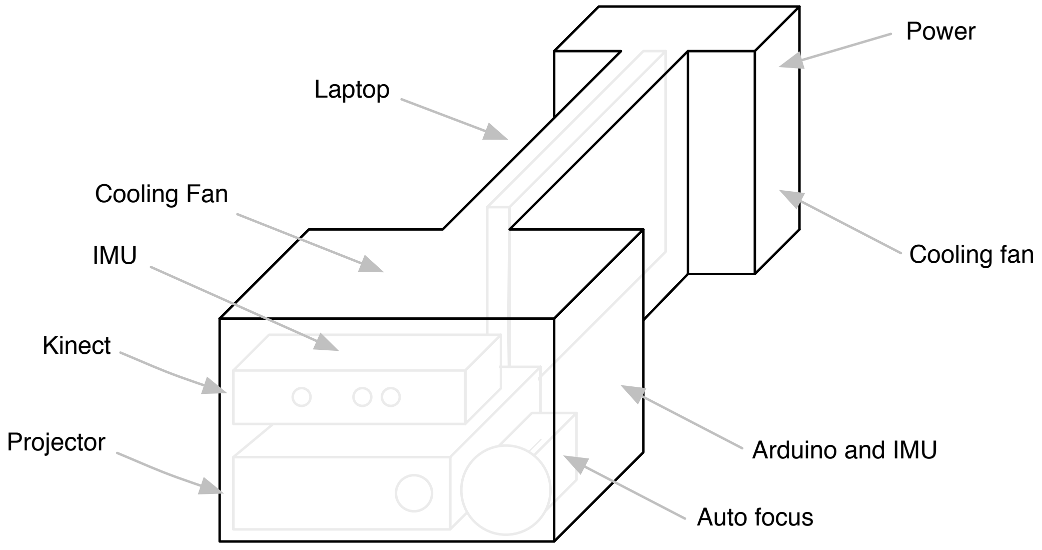


Figure 6: AC adepters in the rear power pod.

The SLAM algorithm is too slow to run on every frame of point cloud data. To address this, the Arduino is also connected to an IMU (inertial measurement unit, see figure 5), which combines two sensors — an accelerometer and a gyroscope. These sensors run a basic dead reckoning algorithm to supply the system with inferences about the user's movement through space between "official" updates from SLAM. This creates a more fluid navigation experience for the user at the expense of some precision. The final consideration in the enclosure's design is power management. The initial hope was to make the entire thing wireless, but for various reasons this isn't currently feasible. (The various reasons are discussed at length in the *Compromises and Future Improvements* section of this paper.) In an effort to make the cable situation as clean as possible, the power bricks for all of the hardware in the system are contained inside the rear of the enclosure. (see figure 6) A single AC plug on the exterior of the enclosure supplies all of the hardware, allowing for "one plug" setup of the entire system, along with one switch to turn everything off and on at once.  Figure 7 gives an overview of the hardware behind the überbeamer.



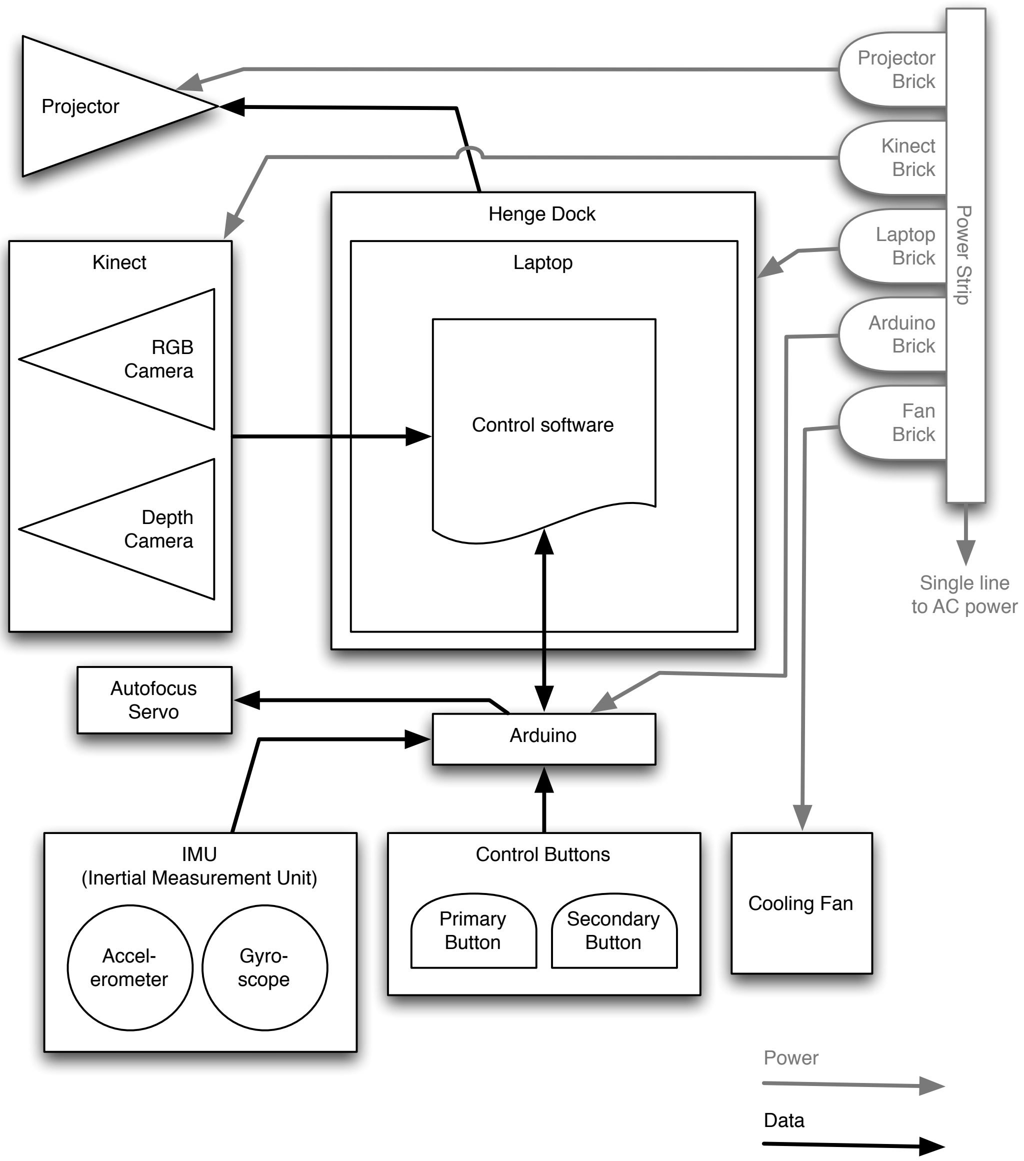


Figure 7: Hardware overview

The enclosure was designed around the dimensions of my 15” laptop computer. The laptop is stored in the “spine” of the enclosure, through a hinged, latching door in the rear of the enclosure. (See figure 8.) Foam padding holds it in place, and all of the connections are made through the henge dock simply by sliding the laptop into the enclosure with some authority. It’s unfortunate that supporting more than one laptop brand / model wasn’t feasible, but the trade-off for not having to fuss with plugging in a tangle of wires each time the laptop is taken in or out of the enclosure was worth it it for the purposes of the prototype überbeamer. (Particularly since I’m likely to be the only user of this prototype.)

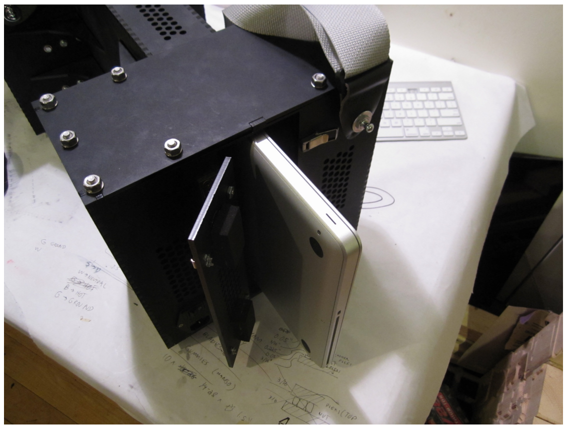


Figure 8: Loading a laptop.

Ergonomics are hit-and-miss. The basic hourglass footprint of the enclosure makes it relatively intuitive to hold the überbeamer at your side, with the projector pod in front of you and the power pod behind your back, keeping the unit’s center of gravity close to your own. This works well, since you end up facing in the direction of the projection. The entire unit is also extraordinarily and unreasonably heavy. In order to realize the one-cable design goal, the user ends up lugging the laptop and all of the AC adapter power bricks along with them. Technically, you would have enough hardware to do intrinsic projection with just the items in the front pod (which holds the the projector, Kinect, Arduino and — but this would mean a bundle of cables, all of which would have to be of significant length if any mobility is to be realized. Both approaches seem mediocre, I opted for the single-cable solution partially by underestimating the total weight of the enclosure, and because managing a bundle of cables is an impediment to the user’s freedom of movement and discourages quick experiments with the system. Subsequent revisions might use a frame-based construction via aluminum support rails to hold everything in alignment instead of the current prototype’s extremely heavy acrylic monocoque construction.

**Software Design**

As of this writing, software development is still very much under way. Building the enclosure was more involved than anticipated, and ended up eating into time schedule for code work. Therefore, some details in this section will be slightly more vague than I would like, or will discuss techniques and decisions that are planned / intended courses of action rather than implemented solutions.

The basic structure of the software systems are shown in figure 9. The software layer can be broken down into two groups, the library / API layer, which is concerned with making intrinsic projection functional

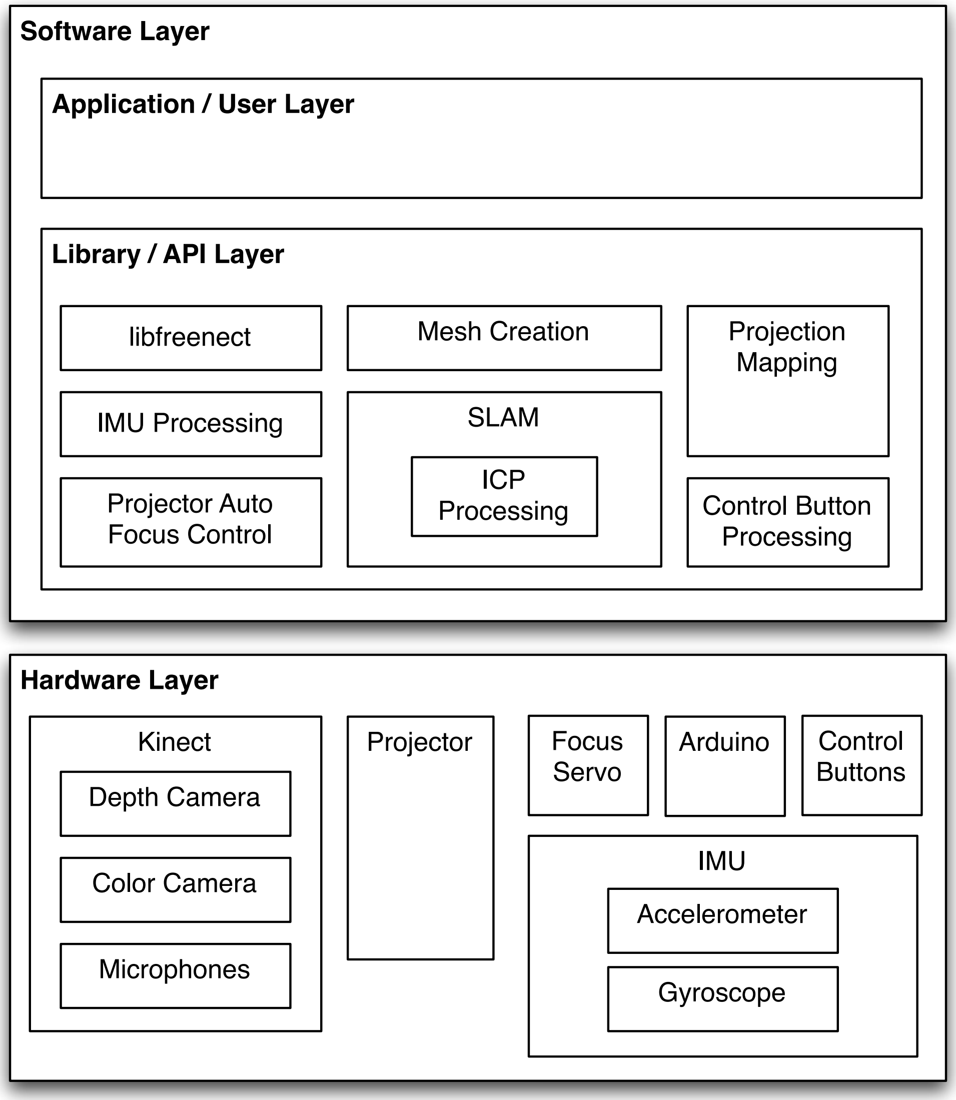


Figure 9: Software overview

and easy to integrate, and the application / user layer, which involves building new interfaces and functionality on top of a working intrinsic projection implementation. The long-term vision for the system is to build a library that abstracts away the configuration and tracking involved in intrinsic projection and makes a dynamic drawing surface available to a range of applications and development environments. In the short term, the ambition is just to get the system working reasonably well in any capacity.

Basic program flow involves taking depth data from the Kinect into openFrameworks via the libfreenect drivers. From there, the depth bitmap is converted from pixel space to world space and perspective correction is applied. Next, depth measurements from the center of the point cloud are checked against a table of stored projector focus values, and a command is sent to the Arduino to turn a servo and keep the projector in focus. Meanwhile, the SLAM algorithm gets to work on the most recent point cloud. It looks at the preceding cloud data and figures out how the latest one needs to be translated and rotated to fit into the larger picture. This is calculated in combination with data from the IMU (gyroscope and accelerometer), which is used to give the SLAM algorithm a "best guess" from which to start looking for ideal fit between point clouds. Likewise, since the SLAM algorithm takes more than the 30 milliseconds or so that we have between frame updates, the same IMU data is used to supply updates regarding the user's position and orientation in space until the latest SLAM update arrives. Finally, mesh generation routines are used to convert the point clouds into polygons for rendering in OpenGL. Generating meshes from point clouds is a non-trivial problem, but the solution I'm using involves placing the user in a spherical mesh which is subsequently "stretched" by the point cloud data and subdivided as necessary to keep resolutions reasonable. This solves some problems (e.g. how to handle unmapped space), but creates others which will be discussed below. Using the world-space position information derived from SLAM (the"localization" part), we can position a virtual camera in our model of the room using OpenGL, matching the focal length and position in such a way that the pyramid of light thrown by the projector in real space matches the pyramid of vision seen by the camera in OpenGL's virtual space. This provides projection mapping / keystone compensation to keep the real and virtual worlds aligned. Finally, pixel content is drawn as texture data into the polygonal mesh of the space. If a user is actively drawing into the space or interacting with a dynamic interface element, then perspective-compensation might be applied to make content projected on surfaces oblique to the user appear correct, even if the content would appear distorted to someone standing perpendicular to the wall. (For example, someone who calls up a circular menu on a wall at a steep angle to their position would actually draw a stretched circle that would look correct from where the user's standing but incorrect from any other.) It might be necessary to force certain tool functions to snap to the normals of the surface to avoid slight perspective errors from accumulating.

SLAM and other spatial reconstruction algorithms are usually able to perform loop closure, which is the matching, alignment, and subsequent correction of point cloud data between clouds that were not initially captured in close proximity. Put more simply, imagine you're mapping a simple, square room. You start at the door and work your way clockwise around the room, using SLAM to build a map along the way. By the time you're coming back to where you've started, error has accumulated in the SLAM model and it's likely that the most recent pieces of wall you've filled into the model is slightly out of alignment with the very first section of wall modeled when you started. Loop closure involves recognizing these instances and, as its name implies, closing the loop programmatically.  This turns out to be a very challenging problem, and most solutions involve feature detection techniques that take place in RGB space and not in depth space, which could present feedback problems with the überbeamer's projection (and reduces the viability of working in very low light environments). Instead of dealing with this, the überbeamer uses the spherical mesh generation technique described above, where the user starts in an a mesh that's already closed and merely stretches / distorts it as they move. This solves most of the loop closure problem, since it means there are never any open holes to close in the first place, but it creates some problems of its own. For example, what would happen if a user tries to map a toroidal space? Under what conditions should the user be able to punch through the mesh? These are the kind of problems for which solutions will have to evolve as our understanding of the unique demands of intrinsic projection improves.

**Research and Influences**

The überbeamer's form factor was inspired by a previous project I created for NIME in the fall of 2010. This project, called the VHS Can[n]on, took a VCR and turned it into a wearable instrument for sampling audio-visual loops from VHS cassettes in real time. (See figure 10.) The Can[n]on was played by pointing the VCR towards a projection screen and playing / stopping VHS tapes. The absurdity of gesturing with and wearing an appliance was surprisingly appealing. However, the system was constrained by the static nature of the projector. Regardless of where you moved during a performance, you were still limited to the projector's stationary and relatively small rectangle of light. This sense of claustrophobia seeded the interaction model proposed in my thesis, in which the projector itself is handheld and any surface in a room becomes a canvas for projection. It's also easy to imagine more potential applications for this platform than the VHS Can[n]on alone, so I decided to build the überbeamer as a content-agnostic chain of open-source hardware and software tools instead of a singular project. As such, the überbeamer is "an interface" rather than "the interface", and it's my hope that its most interesting applications have yet to be realized

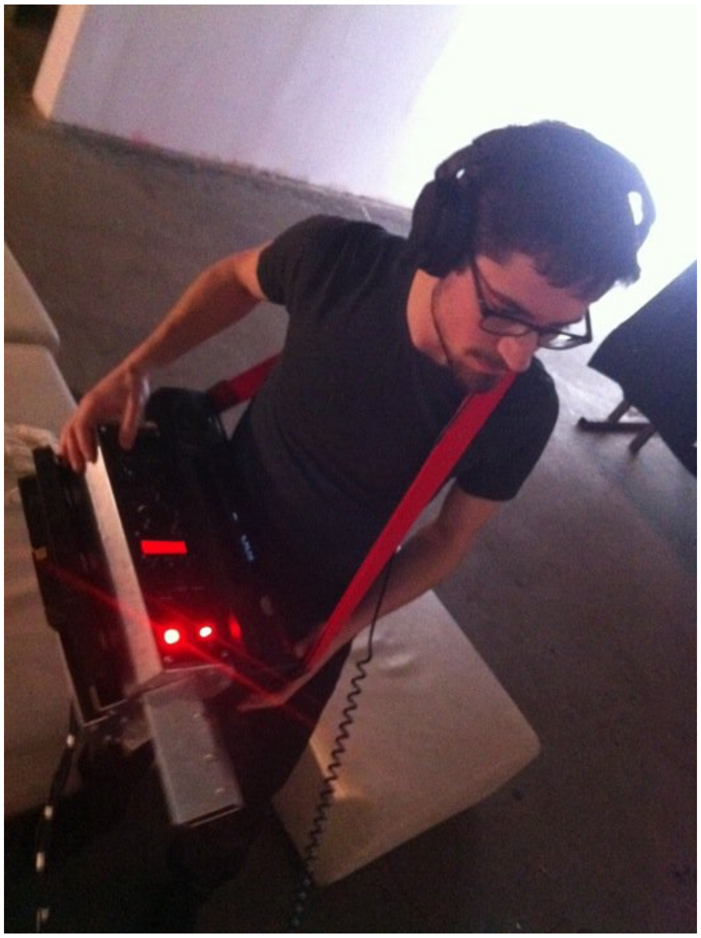


Figure 10: VHS Can[n]on

With about a month remaining in the thesis production cycle, a fellow ITP student brought a project by Xiang Cao et al. from

2005 to my attention. Cao's work on handheld spatial projection mapping represents exactly the interaction model I had in mind for the überbeamer. This news came too late for development of a completely new thesis, so I was forced to give up the notion that the überbeamer presented a completely original and novel mode of interaction and instead emphasize the ways in which recent technology allows for improvements on Cao's approach. Specifically, Cao's implementation depends on an external camera to track optical markers on a small projector in combination with a hand-built three dimensional model of the projection space. My implementation uses the Kinect + SLAM / ICP algorithms interleaved with positioning inferences from an IMU to allow for real-time construction and projection mapping in unknown environments without external hardware dependencies. Furthermore, Cao did not explore possibilities surrounding surface ingestion… e.g. use of the Kinect's RGB camera to grab chunks of the environment for processing / re-projection. Finally, the überbeamer allows for portability. The entire design is a single unit slung over a shoulder. It won't be as compact as Cao's hand-held projector since the Kinect is bulky and a powerful laptop is required to run the mapping algorithms. Finally, the überbeamer brings the possibility of location tagging / persistent content. Again, because of portability and the relative lack of calibration required for extracting positioning info from a point cloud history, it would be possible to save layers of projected content that are linked a specific location and could be edited / overwritten on subsequent visits to the space.

**Compromises and Future Improvements**

In the course of designing the überbeamer, several aspects of the hardware design and software had to be compromised or otherwise constrained in order to maintain a reasonable schedule and scope for the project. The first and perhaps largest compromise was the decision to make the hardware wired instead of wireless. From my experience with the VHS Can[n]on, it was clear that the set-up time involved in connecting an unwieldy bundle of cords makes using a device a real production. This discourages spur-of-the moment experimentation and slows the software iteration process.

At the moment, the deciding factor in the wireless vs. wired decision process is the projector. The current state of pico projector technology is such that units with power demands low enough to be run off of a battery of reasonable size and weight are simply too dim and small for the überbeamer's intended application. (As of this writing, a state-of-the-art consumer pico projector like the 3M MP160 emits only 32 lumens and tops out around 80" diagonal. In comparison, the Optoma TX330 emits 2200 lumens and is good for a 302" projection.) Because of the high power demands (225 watts) of the TX330, battery power is out of the question. However, in the interest of avoiding the cord tangles and long set-up times and bulky external hardware of the VHS Can[n]on, the überbeamer's enclosure is designed to carry everything it needs to operate, so the only external connection required is a single AC power cable. The trade-off is a significant weight increase to accommodate a laptop, power bricks, etc., and every extra pound means the überbeamer will be more difficult to maneuver and more fatiguing to use.

Another initial hope for the überbeamer involved operation outdoors. Unfortunately, the Kinect is extremely sensitive to IR and doesn't work well at all in daylight. This means operating the überbeamer at dusk or even in rooms with large amounts of incident sunlight is not possible. The upside, however, is that the Kinect works formidably in low light situations, so projecting outdoors at night should not be an issue.

The Kinect also limits possibilities for multi-user interaction. Although building more than one überbeamer is beyond the scope and time constraints of an ITP thesis project, it still would be nice to consider the possibility of several people working with their own überbeamers in the same space with the same set of data. Interference between the Kinect's infrared dot patterns is likely to make this an impossible scenario.

Finally, running the SLAM algorithm responsible for the überbeamer's real-time reconstruction of three dimensional space is extremely processor intensive. A laboratory full of Intel research scientists only managed to get the algorithm down to about 100ms per frame. To convey fluid movement through a space, the algorithm would need to run about three times as fast (and that's not even considering cycles required for the rest of the application's functionality). Assuming that this magnitude of improvement is possible on an algorithm that has already been optimized by experts would be foolhardy. So, instead of spending time optimizing the algorithm, I opted to use an IMU to provide inferences about spatial movement between SLAM updates to maintain the illusion of fluid movement through the space. This introduces more glitches into the user experience than I would like, but it's a necessary shortcut to get the system working at all within the time available — for now, fluidity trumps accuracy.

Text entry is another issue that goes unaddressed in this version of the überbeamer. I went with the two-button, mouse-like interaction approach to keep things as simple and intuitive as possible. Text entry would require something more elaborate — either a physical keyboard, a projected keyboard, voice recognition, or some kind of gestural system in the spirit of Palm's Graffiti system. These are interesting usability challenges, but are beyond the scope of a first iteration of the platform.

## Conclusions

On a long enough timeline, each of the present compromises should prove surmountable. Pico projector brightness and power consumption will improve. Cameras providing Kinect-like depth maps will become more compact. CPU speed improvements and other optimizations will shave milliseconds off of the SLAM algorithm until running in true real time is possible. Until then, the überbeamer is stuck with physical bulk bordering on absurdity, and a range-limiting tether to line power. For those who would rather not wait for practicality to catch up with their interests, the überbeamer can fill their niche with an open-source suite and an open hardware design for exploring the possibilities of intrinsic projection.

---

## Additional Information and Updates

Ongoing work on the überbeamer project will be documented on my website, *uberbeamer.com*. Additional resources will be linked there. Source code is available at *github.com/kitschpatrol/uberbeamer*.

## Acknowledgments

Thanks to my thesis advisors, Nancy Hechinger and Marina Zurkow. Thanks also to my peers in the redbird and bluebird thesis development groups. My thanks to Dan Shiffman for his help with the ICP algorithm. I am indebted to the open-source creative coding community, particularly the Processing and openFrameworks projects.

**Works Cited**

Apple Computer, Inc. *Macintosh Human Interface Guidelines*. New York: Addison Wesley, 1992. Print.

IBM Research. "The Everywhere Displays Project." <http://www.research.ibm.com/ed/>. Web. 11 April 2011.

Gruber, John. "Leaked Screenshots of Microsoft Office for Mac 2011." Daring Fireball. 30 March 2010. <http://daringfireball.net/linked/2010/03/30/office-mac>. Web. 11 April 2011.

McCall, Anthony. *The Solid Light Films and Related Works*. Ed. Christopher Eamon. Evanston, IL: The Northwestern University Press, 2005. Print.

O'Shea, Chris. *Out of Bounds*. 2007. Design Museum, London. <http://www.chrisoshea.org/out-of-bounds>. Web. 11 April 2011.

Sobecka, Karolina. *Wildlife*. 2006. Brooklyn, NY. <http://www.gravitytrap.com/artwork/wildlife>. Web. 11 April 2011.

Tsubokura, Teruaki. *Shadow Touch*. Date unknown. Location unknown. <http://www.youtube.com/watch?v=sZ8fbp4b3Jw> Web. 11 April 2011.

Xiang Cao, Ravin Balakrishnan. (2006). Interacting with dynamically defined information spaces using a handheld projector and a pen. *Proceedings of UIST 2006, ACM Symposium on User Interface Software and Technology*. p. 225-234.

Xiang Cao, Clifton Forlines, Ravin Balakrishnan. (2007). Multi-user interaction using handheld projectors. *Proceedings of UIST 2007, ACM Symposium on User Interface Software and Technology*. p. 43–52

Xiang Cao, Jacky Jie Li, Ravin Balakrishnan. (2008). Peephole pointing: Modeling acquisition of dynamically revealed targets. *Proceedings of CHI 2008, ACM Conference on Human Factors in Computing Systems*. p. 1699–1708.