Middlebury

Support through NSF Grant IIS–1718376 is gratefully acknowledged

Generating Benchmark Datasets for Computer Vision

Daniel Scharstein
Charles A. Dana Professor of Computer Science

Toby Weed
Middlebury class of 2022

Motivation

Computer vision is one of the most active research areas in computer science: think self-driving cars, facial and object recognition, image classification, and beyond.

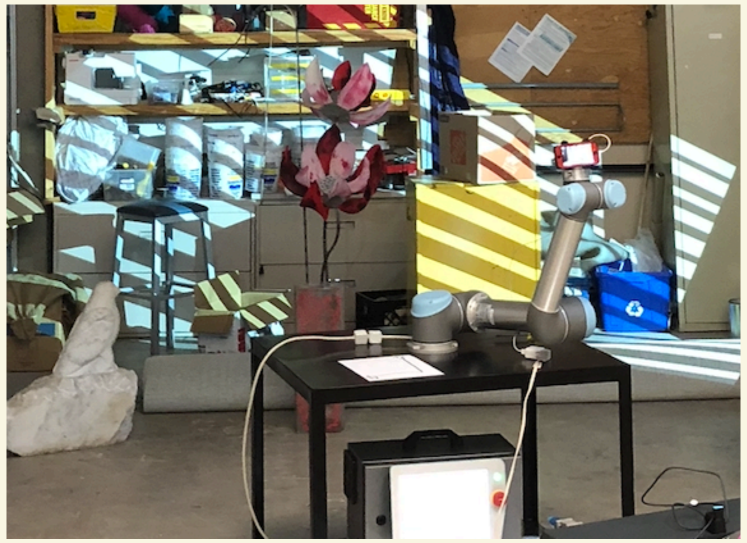
In order to facilitate the development of better computer vision algorithms by hundreds of different teams across the world, it is necessary to know what "better" means.

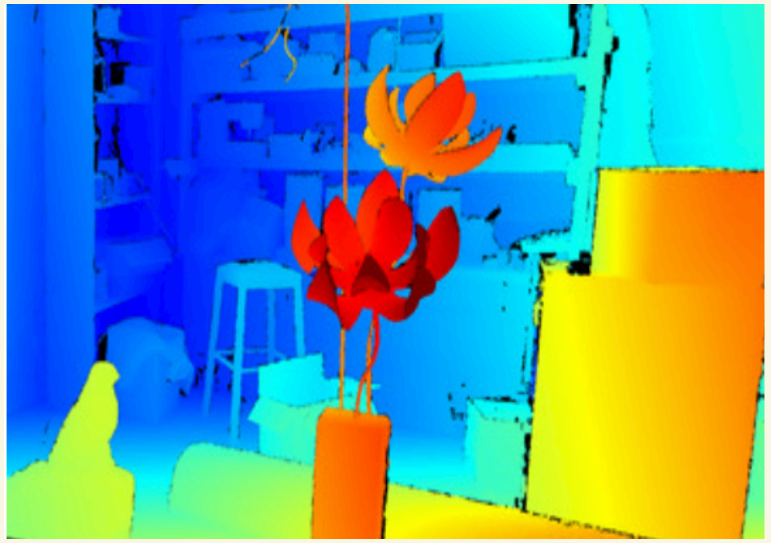
The purpose of our research is to produce datasets which will help set the bar for computer vision algorithm performance.

Methods: Researchers will run their algorithms to extract depth information from our scenes, and we'll compare their results with our own. Thus, a large part of our work is **extracting highly accurate 3D scans from pictures.**

Ground truth & Structured Lighting

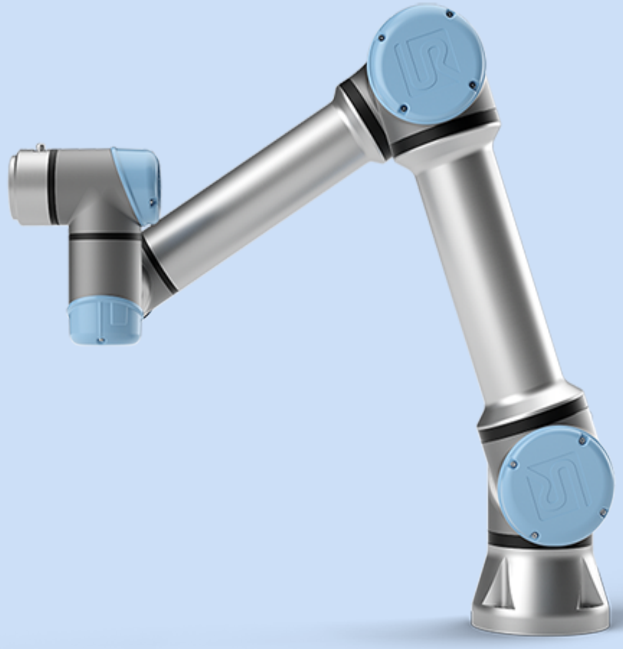
To get a highly accurate ground truth depth map of a scene, we project unique pixel codes on it and compare their motion.





Robotic Viewpoint Control

We use a UR5 industrial robot arm for repeatability and precision.



Summer Progress & Future Work

This is the fourth summer of research on this dataset, building on code from past interns. Here's what we worked on:

- **Replacing C++ calibration code** with a more flexible module, including a live visualization GUI.
- **Refining Swift, C++, & Python control servers** to expedite future maintenance, handover, and scene capture.
- **Reimagining the end-goal** to best serve rapidly evolving evaluation needs.

We're hoping to continue moving forward during the academic year by integrating **realistic motion tracking** and **capturing production datasets**.

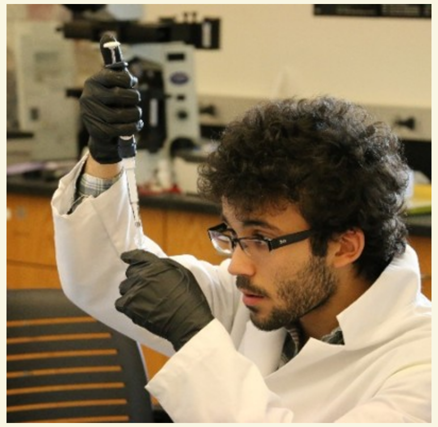
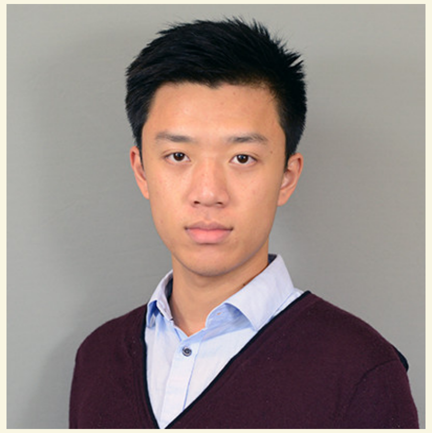
Background & Future

This project has been worked on by numerous interns in the past:

- Guanghan Pan ’21, Tiansheng Sun ’20, me (summer ‘19)
- Nicholas Mosier ’20, Tommaso Monaco ’20, Roger Dai, ’20 (summer ‘18)
- Nicholas Mosier’20 (summer ‘17)

I am continuing work this fall, but I expect (Professor Scharstein can confirm or deny) that more interns will be needed in summer ‘21).

.

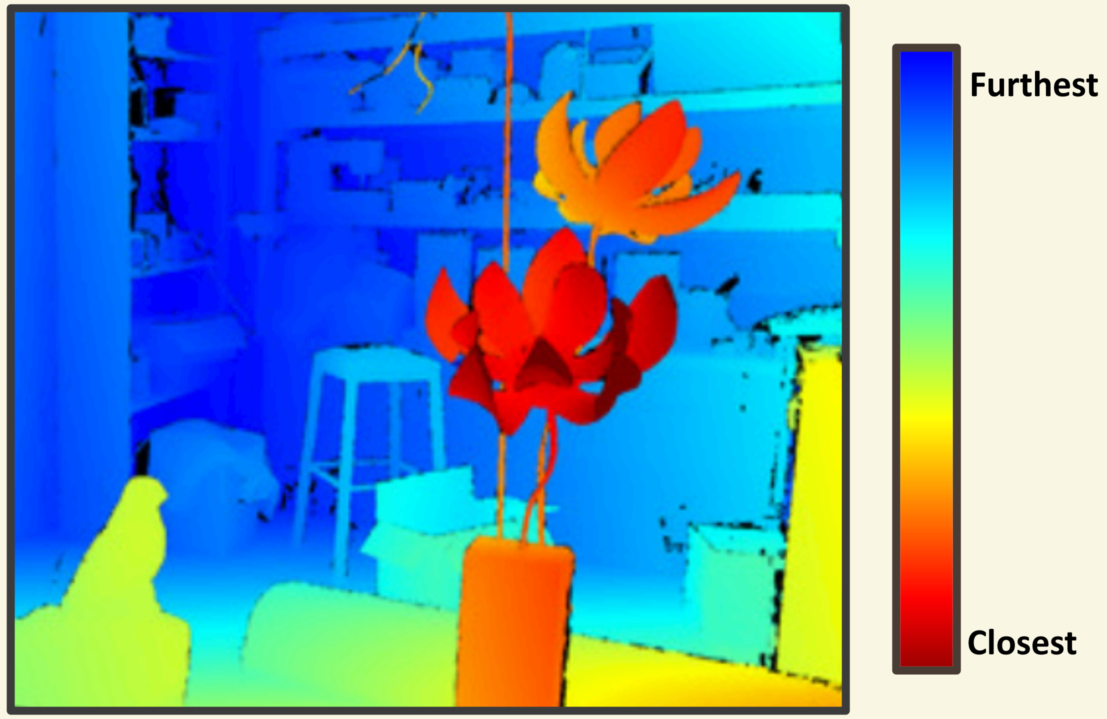



Motivation

Progress in stereo vision depends on the availability of precise methods for evaluating the accuracy of different algorithms.

This project will fill that void by providing diverse datasets with subpixel accurate ground truth depth maps against which researchers will be able to compare the results of their algorithms.

So how do we get from here…

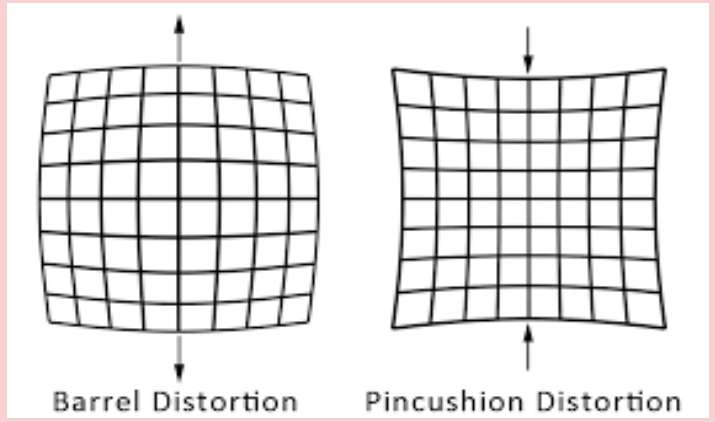


…to here?

Example Use Cases for Mobile-Based Depth Perception

- **3D scanning of objects or faces:** e-commerce, Snapchat filters, security cameras.
- **Photo enhancement**: portrait mode, automatic editing effects.
- **AR/VR**: VR headsets, gaming, augmented video streaming.

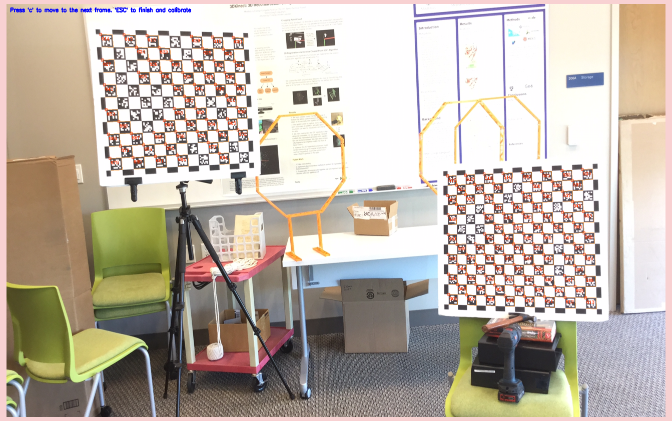
Step One: Calibration



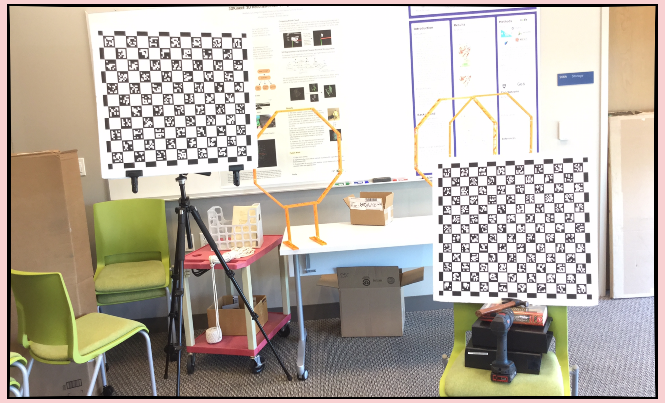
Lens Distortion

Solved by $\longrightarrow$

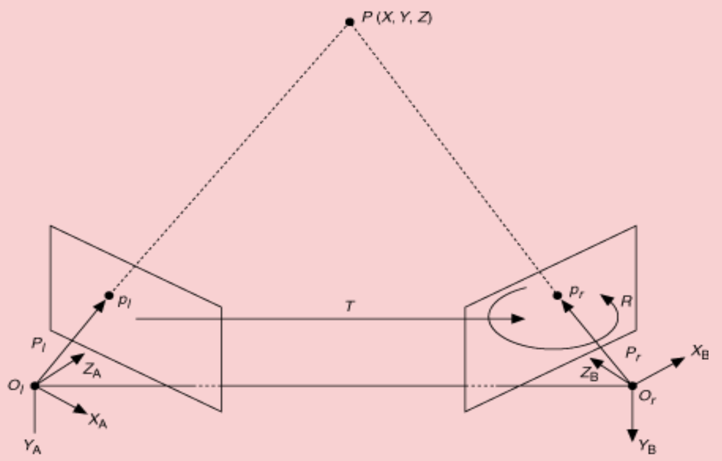
Intrinsic Calibration


Press 'c' to move to the next frame. 'ESC' to finish and calibrate

Marker Detection



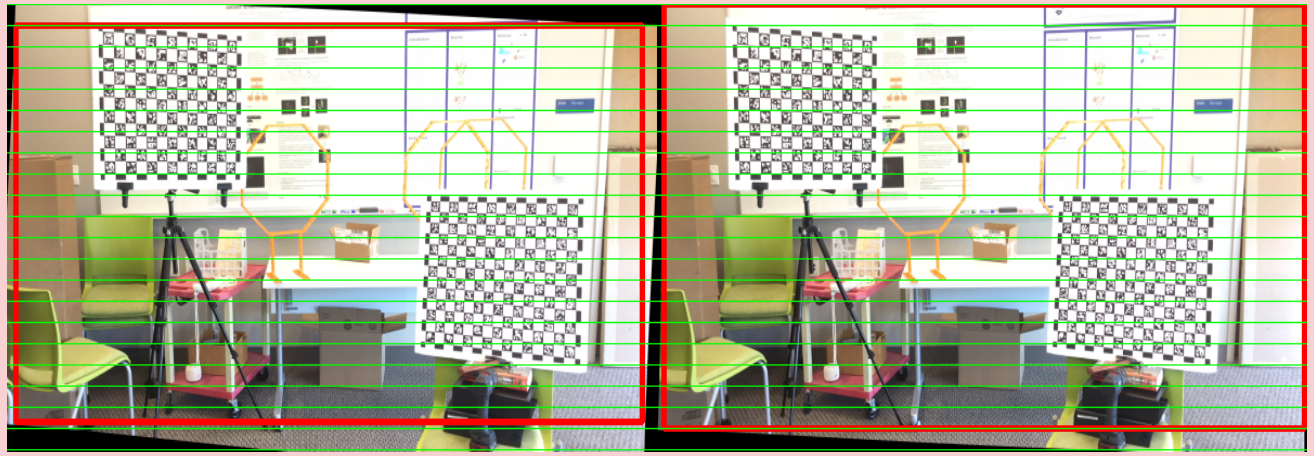
Undistortion



Viewpoint Disparities

Solved by $\longrightarrow$

Extrinsic Calibration



Rectification

Every camera lens causes some distortion, which we must account for if we seek to construct accurate 3D models. Additionally, we want to project images from each viewpoint onto a common plane to make finding their view disparities easier, which requires finding their locations & orientations relative to each other.

Step Two: Structured Lighting & Disparity Matching

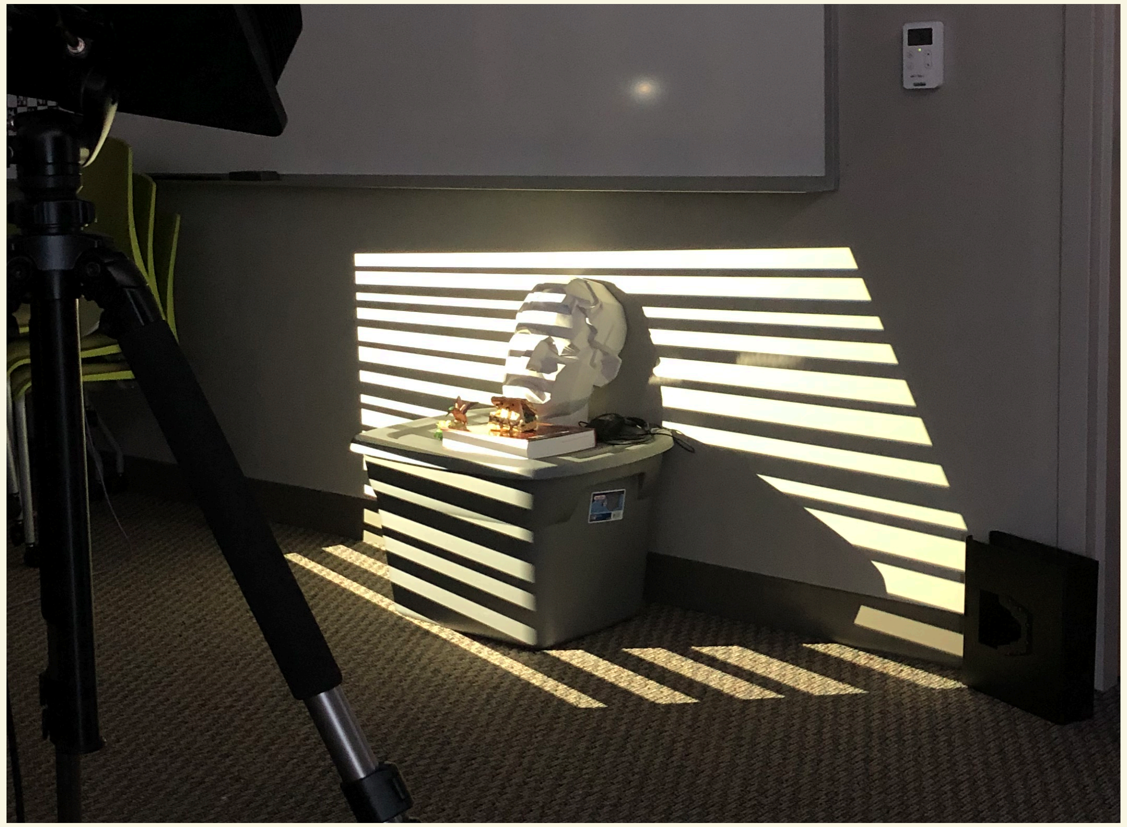
Stereo Vision

Stereo vision is the extraction of 3D information from multiple 2D images taken from different locations. Our eyes do it and so can computers.

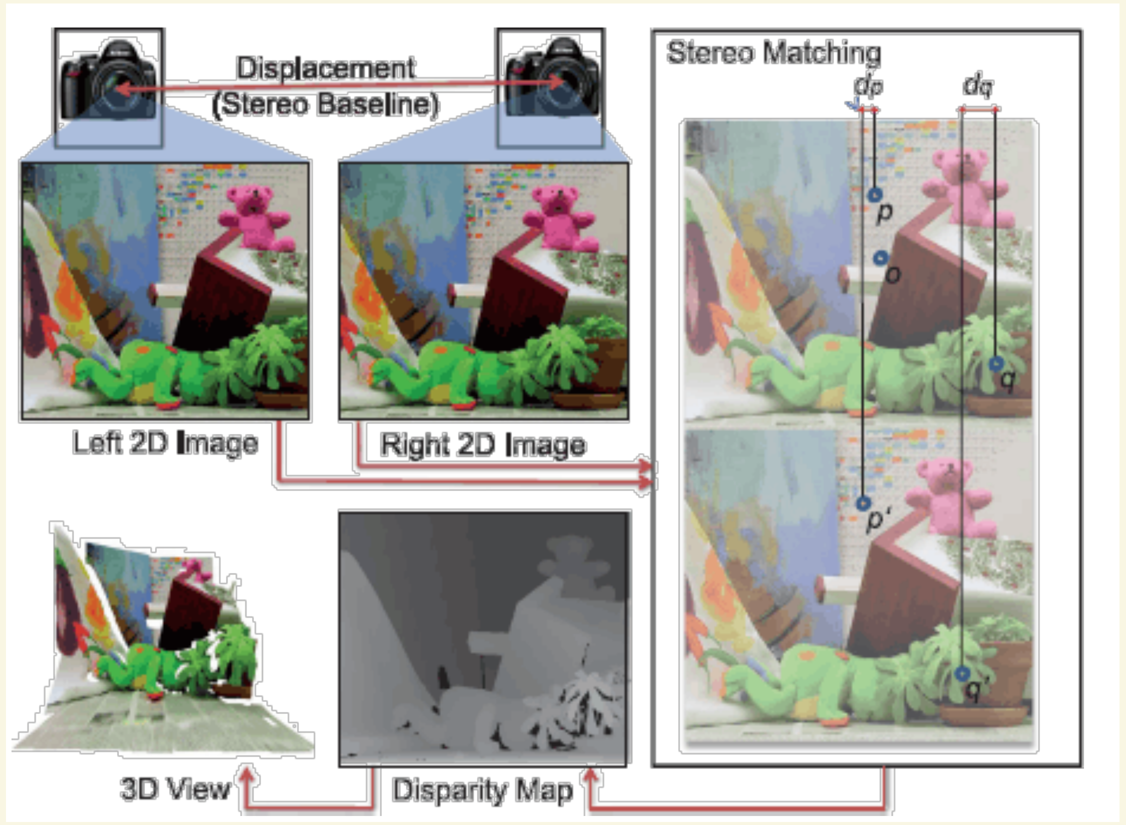
Structured Lighting

The special sauce. By projecting structured lighting displays onto a scene, we can assign a **unique code to each projector pixel**, compare the movement of pixels between different viewpoints, and thus determine the proximity of a surface.

More movement means the surface is closer, and less means it's farther.



Projecting structured lighting onto a scene to obtain unique pixel codes for each point on a surface. We project 20 patterns, 10 with vertical stripes and 10 horizontal, to get a unique code for each projector pixel.



Comparing 2D codes of points on image to extract depth information.

Step Three: Image Processing

To get ground truth depth maps of the highest possible quality, we use an image processing pipeline with 6 steps:

Decode images
Decode the structured lighting images to get a pixel code for each point in the scene. The projected patterns act as a binary code for the X and Y coordinates.

Rectify images
Using the extrinsic calibration parameters generated from calibration, rectify images from adjacent camera views so that they appear to lie on the same plane.

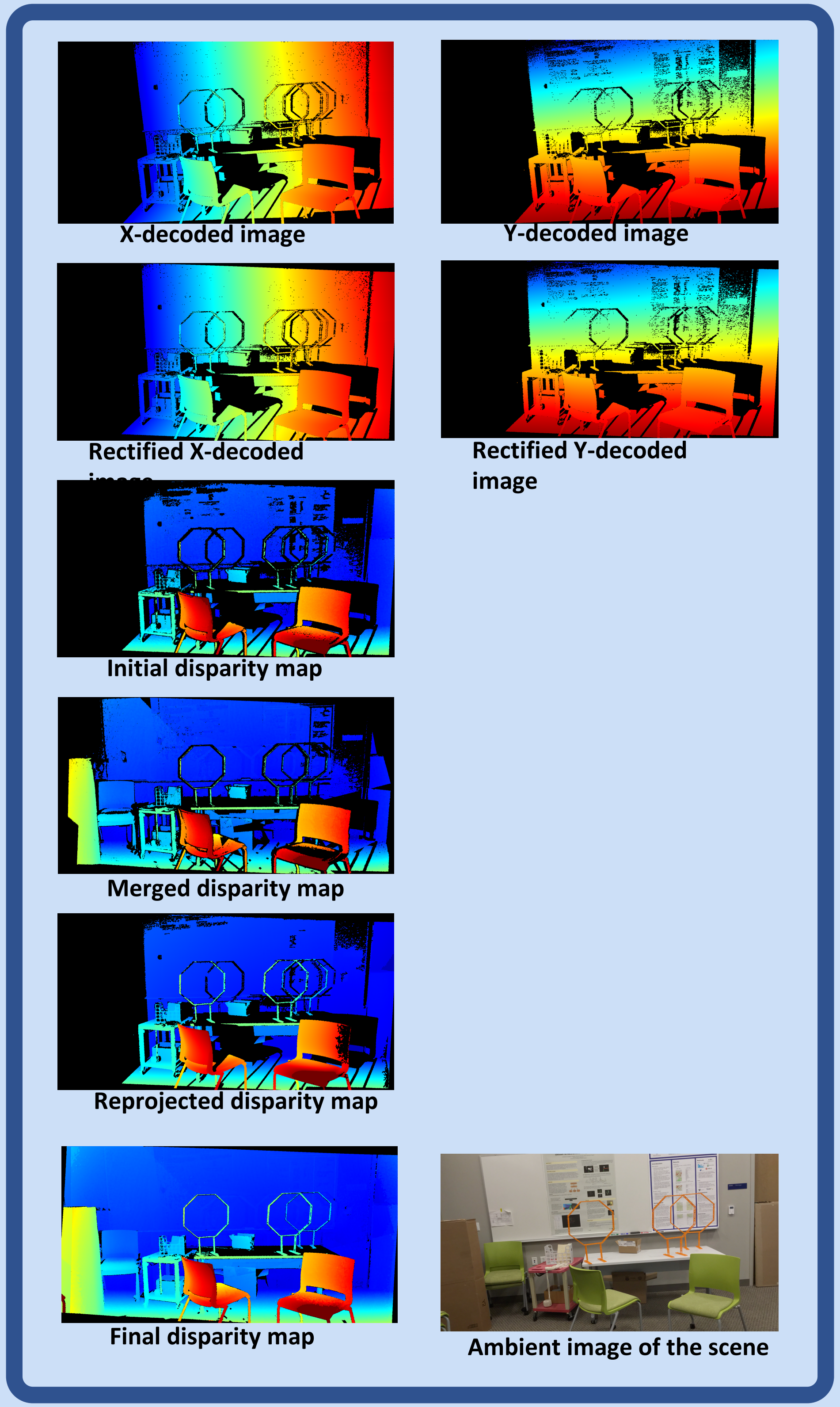
Disparity match images
Merge the X and Y decoded images to get a unique pixel code for each point. Then compare the pixel codes from two views to get a disparity map.

Merge disparity maps
Merge disparity maps produced by different projectors to fill in as many depth values as possible as accurately as possible.

Reproject merged disparity maps
The merged disparities are used to self-calibrate the projector positions. Once the projector relationships are known, half-occluded regions can be filled in with disparity values.

Merge2
Put together original disparities, merged disparities, and reprojected disparities to fill in all possible values.

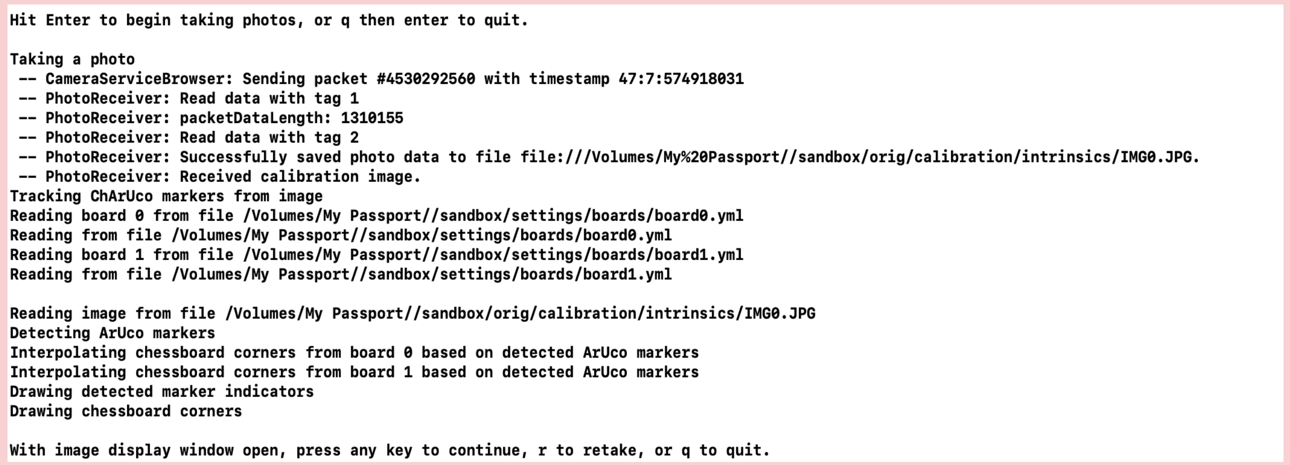
These steps are all implemented using C++ in MobileLighting Mac.


X-decoded image
Y-decoded image
Rectified X-decoded image
Rectified Y-decoded image
Initial disparity map
Merged disparity map
Reprojected disparity map
Final disparity map
Ambient image of the scene

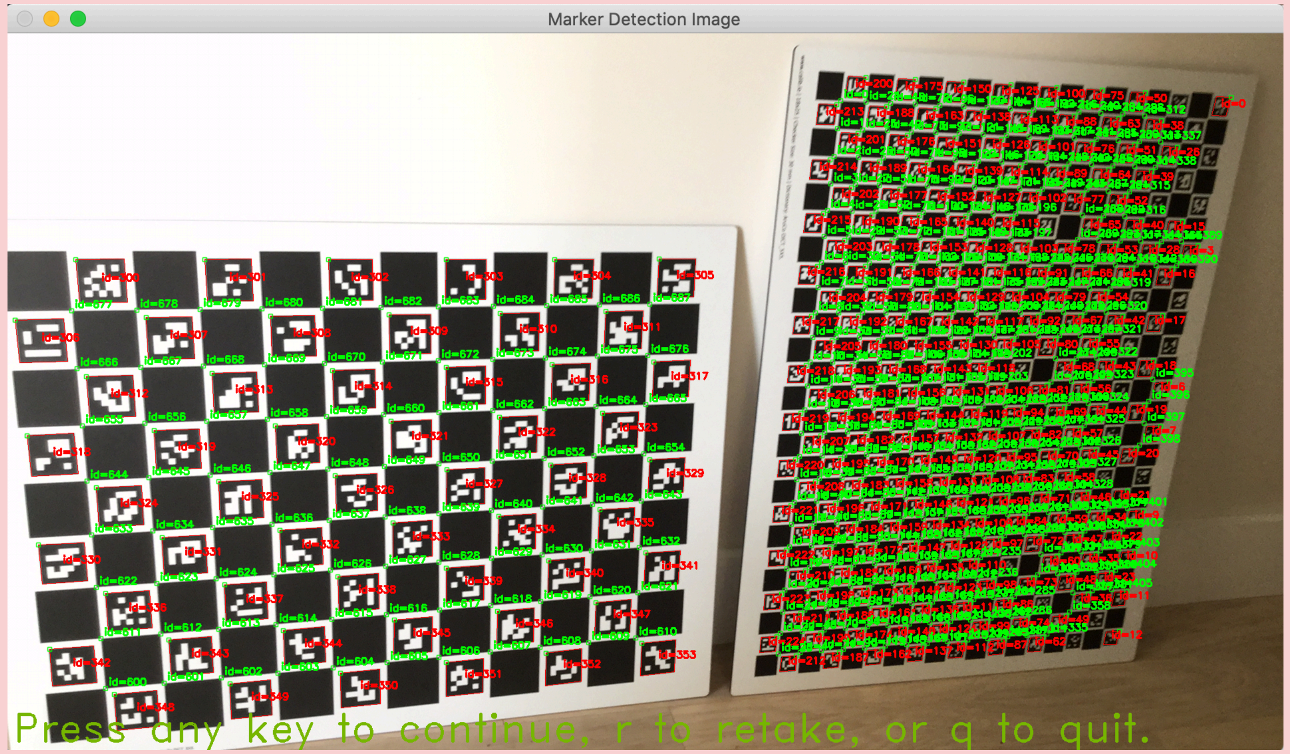
The image processing pipeline

Contributions this summer

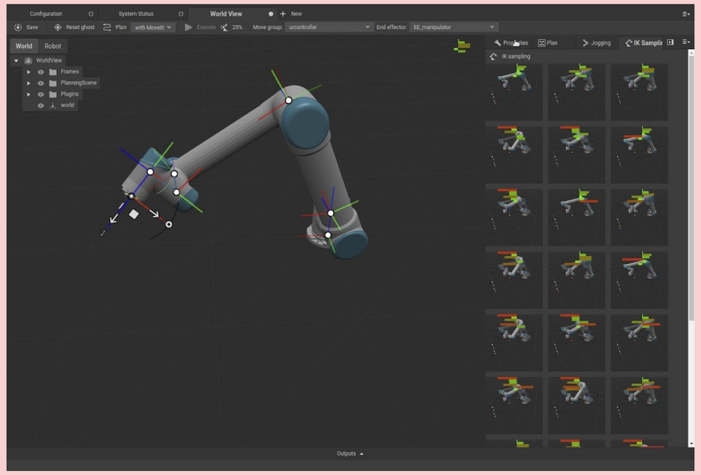
Marker Detection Visualization & Calibration Refactor



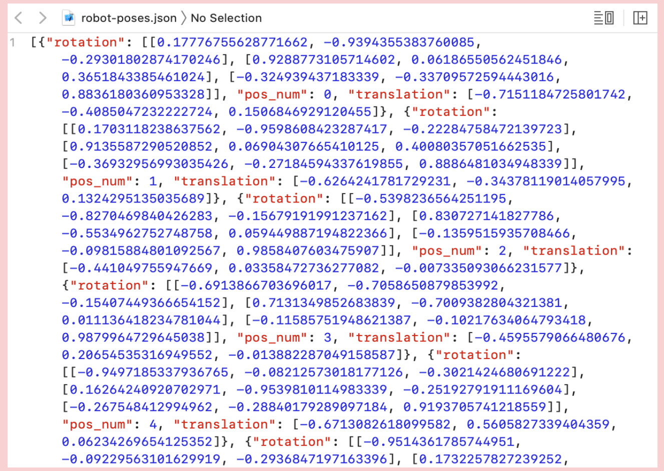
```
Hit Enter to begin taking photos, or q then enter to quit.

Taking a photo
 -- CameraServiceBrowser: Sending packet #4530292560 with timestamp 47:7:574918031
 -- PhotoReceiver: Read data with tag 1
 -- PhotoReceiver: packetDataLength: 1310155
 -- PhotoReceiver: Read data with tag 2
 -- PhotoReceiver: Successfully saved photo data to file file:///Volumes/My%20Passport//sandbox/orig/calibration/intrinsics/IMG0.JPG.
 -- PhotoReceiver: Received calibration image.
Tracking ChArUco markers from image
Reading board 0 from file /Volumes/My Passport//sandbox/settings/boards/board0.yml
Reading from file /Volumes/My Passport//sandbox/settings/boards/board0.yml
Reading board 1 from file /Volumes/My Passport//sandbox/settings/boards/board1.yml
Reading from file /Volumes/My Passport//sandbox/settings/boards/board1.yml

Reading image from file /Volumes/My Passport//sandbox/orig/calibration/intrinsics/IMG0.JPG
Detecting ArUco markers
Interpolating chessboard corners from board 0 based on detected ArUco markers
Interpolating chessboard corners from board 1 based on detected ArUco markers
Drawing detected marker indicators
Drawing chessboard corners

With image display window open, press any key to continue, r to retake, or q to quit.
```



Universal Robot State Tracking



MLRobotControl server (Python)



[{"rotation": [[0.17776755628771662, -0.9394355383760085,
    -0.29301802874170246], [0.9288773105714602, 0.06186550562451846,
    0.3651843385461024], [-0.324939437183339, -0.33709572594443016,
    0.8836180360953328]], "pos_num": 0, "translation": [-0.7151184725801742,
    -0.4085047232222724, 0.1506846929120455]}, {"rotation":
    [[0.1703118238637562, -0.9598608423287417, -0.22284758472139723],
    [0.9135587290520852, 0.06904307665410125, 0.40080357051662535],
    [-0.36932956993035426, -0.27184594337619855, 0.8886481034948339]],
    "pos_num": 1, "translation": [-0.6264241781729231, -0.34378119014057995,
    0.1324295135035689]}, {"rotation": [[-0.5398236564251195,
    -0.8270469840426283, -0.15679191991237162], [0.830727141827786,
    -0.5534962752748758, 0.059449887194822366], [-0.1359515935708466,
    -0.09815884801092567, 0.9858407603475907]], "pos_num": 2, "translation":
    [-0.441049755947669, 0.03358472736277082, -0.007335093066231577]},
    {"rotation": [[-0.6913866703696017, -0.7058650879853992,
    -0.15407449366654152], [0.7131349852683839, -0.7009382804321381,
    0.011136418234781044], [-0.11585751948621387, -0.10217634064793418,
    0.9879964729645038]], "pos_num": 3, "translation": [-0.4595579066480676,
    0.20654535316949552, -0.013882287049158587]}, {"rotation":
    [[-0.9497185337936765, -0.08212573018177126, -0.3021424680691222],
    [0.16264240920702971, -0.9539810114983339, -0.25192791911169604],
    [-0.267548412994962, -0.28840179289097184, 0.9193705741218559]],
    "pos_num": 4, "translation": [-0.6713082618099582, 0.5605827339404359,
    0.06234269654125352]}, {"rotation": [[-0.9514361785744951,
    -0.09229563101629919, -0.2936847197163396], [0.1732257827239252,

Trackfile for ML_Mac system coordination server (Swift & C++)

Full-System Usability, Modularity, Style Upgrades

- Integrated processing mode for detachability
- Improved usage messages
- Fixed bugs
- Improved package management and portability
- Standardized input and output files

Challenges

- Large, unwieldy codebase
- Tenacious iPhone and Mac communication issues
- Inability of robot to capture smooth video
- Mixed languages, protocols, and programming paradigms

References and Credits

Learn More

Middlebury Stereo Homepage:

http://vision.middlebury.edu/stereo/

Auto-generated web pages displaying some samples from datasets we took last summer:

shorturl.at/gEMRZ

shorturl.at/dBGMQ

Project GitHub (contains tons of information and links to other relevant things):

https://github.com/tobyweed/MobileLighting

Acknowledging to prior Middlebury research assistants, without whom the system wouldn't exist:

- Guanghan Pan '21
- Tiansheng Sun '20
- Nicholas Mosier '20
- Tommaso Monaco '20
- Roger Dai, '20

Thank you to Professor Scharstein, the CS department, and the NSF!