



May 25th, 2021.

Marker Localization with a Multi-Camera System IEEE Paper

Motion Tracking

- Many applications: virtual studios, robotic swarms, human surveillance tracking, wildlife tracking, etc.
- Two main directions (for camera Field Of Views):
 - few overlapping FOV: used for maximized system coverage in video surveillance
 - high redundancy in FOVs: 3D reconstruction, high positioning accuracy/timing
- Most common approach:
 - stereo vision system: uses 2 cameras mounted on a common frame

May 26th, 2021.

CRL Lab Meeting: Research Updates

Did some research on how ArUco markers work and implemented them in a notebook exercise to discuss the research process in this week's CRL lab meeting.

Marker Detection

- Lay **foundations** for **multi camera calibration** and **setup w/ priyanka**.
- **Mainly pose estimation**: very important for computer vision applications (ie. robot navigation, augmented reality, etc.)
- This **process I'm investigating** is based on finding **correspondences** between **points** in the **real environment** and used to **measure** the **position** in **3D** space when information about the camera lens and the size of the marker are known.
- Usually difficult, so OpenCV has **ArUco markers** to make it easier.

An **ArUco** marker:

a type of QR code w/ a **synthetic square marker & wide black border** and an **inner binary matrix** which maps to an **identifier** (id).

- **black border** = facilitates its fast detection in the image
- **binary codification** = allows its identification and the application of error detection and correction techniques

Given an image containing ArUco markers: the detection process has to **return a list of detected markers**.

Each detected marker includes:

- The **position of its four corners** in the image (in their original order).
- The **id of the marker**.

The **marker detection process** is comprised of **two main steps**:

Detection of marker candidates:

- The **image** is **analyzed** in order to **find square shapes** that are **candidates** to be **markers**.

After the candidate detection:

- **determine if** they are **actually markers** by **analyzing** their **inner codification** by **extracting** the **black** or **white bits**.
- The bits are then **analyzed** to determine if the marker belongs to a **specific dictionary**.
- **Output**: to return a **list of detected markers**.

ArUco dictionary:

- specifies the type of ArUco marker we are generating and detecting.
- without the dictionary, unable to generate and detect these markers.

Cool pose detection and tracking computer vision tool: ie. make a [drone landing pad](#) w/ these arUco markers to track the robot's position in the global coordinate system w.r.t the landing pad

Current Steps:

1. Load input image
2. specify and load the appropriate ArUco dictionary
3. create/define the parameters to the ArUco detector (which is typically just a single line of code using the default values)
4. apply the cv2.aruco.detectMarkers to actually detect the ArUco markers in the image or video stream

Two Explored Scripts:

- One Python script: **detect ArUco markers in images (will be easier to start off with b/c still unsure of Google Collab's Live Video Capturing Capabilities)**
- Another: to **detect ArUco markers in real-time video streams**

Process Composed of 3 programs:

- Generate arUco markers (already established online dictionaries)
- Calibrate camera and saves results to disk
- Load results and tests camera calibration

Next steps, Prepping for:

- Gathering resources + data for **multi camera calibration** and **setup**
- Explore [optical 3D position tracking with OpenCV and ArUco markers scenario](#) with potential BCI applications or virtual tools/AR



Immediate Priorities:

- ArUco Calibration

- State-of-the-art prez (ask prof about IEEE citation for vincent's video)
- Video capturing
- Note on this: [What Is Transformation Matrix and How to Use It](#)
- Read research paper: [Stereo camera system calibration: the need of two sets of parameters](#)
- Continue tutorial: [PyImageSearch Detecting ArUco markers with OpenCV and Python](#)
- [Aruco Tracker/aruco_tracker.py at master · njanirudh/Aruco Tracker · GitHub](#)
- Cool neurotechnology inspiration project: [NormandErwan/ArucoUnity: Bring augmented reality to Unity by tracking Aruco markers in real time](#)

SIFT and SURF debugging resources:

- [SIFT and SURF functions with opencv-contrib-python](#)
- [Why can't I import opencv3 even though the package is installed?](#)
- [OpenCV: Introduction to SIFT \(Scale-Invariant Feature Transform\)](#)

Asked for help during the

Check out matt & kyrel slack messages

June 2nd, 2021.

3D Camera Calibration

Challenges

- **Problem:** pinhole cameras can introduce significant distortions to images.
- **Distortion types:** radial and tangential distortion.

- **Radial Distortion:** straight lines appear curved.

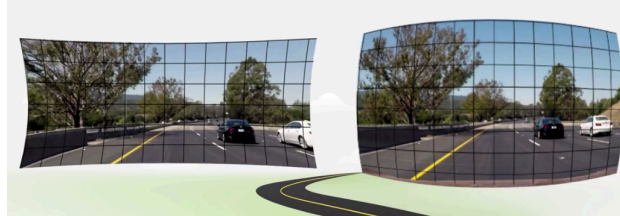
$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

- Represented as:

- Reason:

- Outcome:



- **Tangential Distortion:** heightens when points are farther from the center of the image.

$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$

- Represented as: $y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$

- Reason: image-taking lense is not aligned perfectly parallel to the imaging plane.

- Outcome: some areas in the image look nearer.



- **Solution:** need **distortion coefficients**, **intrinsic** and **extrinsic parameters** of the camera.



- **Distortion Coefficients:**

$$\text{Distortion coefficients} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

- **Intrinsic and extrinsic parameters**

Supplemental State-of-the-art Resources:

- [OpenCV: Multi-camera Calibration](#)
- [OpenCV: Detection of ArUco Markers](#)

- [Fusion Series of Unmanned Sensors-Camera Internal Reference Calibration](#)
- [OpenCV: Camera Calibration](#)
-  Spartan Series / Camera Calibration, Precision + Alignment for ...
-  OpenCV Basics - 16 - Camera Calibration Part 2

June 7th, 2021.

State-of-the-art rehearsal meeting with Priyanka, Quentin, Nada, & Aditi

Feedback about my presentation

- Add slide numbers
- Fix electromagnetic sensing pros and cons (from CSC476 sensory chapter)
- Clarify the function of a laser probe
- Clarify Pose Detection section about why we need arUco markers: “To establish world coordinate system with Aruco markers on the robot’s base”
- Add slide about what I have explicitly done so far in research plan (extra slides portion- good strategy for Q&A)
- Work on intrinsic and extrinsic calibration of images via own camera feed (coding action item)
- Clarify project mission and image

- Too text heavy and technical, decrease it
- Could add animations
- Add title of project
- Have better presentation flow and slide transitions
- Explain any equations shown on slides
- Keep in mind: “Why is this important for others to know about?”
- Could have a scenario story to start (very small medical application)
 - **Key purpose of my ROP:** Need to know where the robot is in space, especially the base in relation to the relevant ROI (region of interest) + cameras (classic application example: in surgical operations, we need to know where the robot is if we’re using any computer vision or automation)
 - robot’s position in space
- Putting too many technical details is confusing and loses people’s understanding + focus
 - Focus on important things like: the context of project, challenges, problem-solving, timeline
 - Add details later on if there is time or add extra slides portion for Q&A session at the very end
- Good presentation timeline order: explain scenario, goal of project (determine location of position of base), different shape sensing, work on specific sensor with camera, camera calibration, why not 2 cameras, 3 cameras, pipeline
- Fix table of contents and match order
- Clarify intrinsic and extrinsic calibration goals (link matlab [diagram](#))
- Add the extra fiducial markers comparison slide to the very end of ppt in case of q & a’s
- Make arUco slides more brief and highlight the main points in the pros and cons list (might not need it at all even- depends on framing)
- Could add a code snippet somewhere (maybe with the arUco sample)

- **Overall perspective:** Like a **funnel** (go from very general -> very specific to pipeline)

June 11th, 2021.

Implementation Tasks: Camera Calibration Phase

Complete by June 17th.

- **Detecting aruco markers** from an image taken from your camera
- Input: Image from either camera , Output: Coordinates in pixels of top left corner
- Implementing **camera calibration** for an available camera (print out a chessboard)

Now that I've gotten a decent grasp over the problem statement, I can start to work on the hands on stuff.

To help you further streamline and focus on the very many tasks, can you start a task list in your journal?

Add checkboxes, and as you work note down subbullets which indicate where youre stuck and the associated date.

this should help populate your journal + help us focus on where the problems are

this will also help you keep track of issues you cant focus on right now - e..g the video capture. So that you can come back to it when needed

Important Dates

Intermediate presentation date: July 14th. 10 min presentation on current progress/results, 5 min Q&A + 5 min feedback

ROP end date: August 16th. Start the documentation + code early.

Final presentation date: August 18th. 15 min presentation + 10 min Q&A + 5 min feedback

Task List

ArUco

- ☒ ~~Detecting aruco markers from an image taken from my camera~~
- ☒ ~~Explore dictionary collections of markers and find optimal one~~
- ☒ ~~[Generate](#) arUco markers~~
- ☒ ~~Print out 6 markers of DICT_6X6_250 (composed of 250 markers and a marker size of 6x6 bits)~~
- ☒ ~~Explore ChArUco boards (besides just ArUco) to see if more accurate marker corners~~
- ☒ ~~Extension: pose detection (+ the x,y,z arrows visual)~~
- ☒ ~~Planar visual for aruco marker detection~~

Camera Setup

- ☒ ~~Explore setup: Input (Image from either camera), Output (Coordinates in pixels of top left corner)~~
- ☒ ~~Implement **camera calibration** for an available camera (print out a chessboard)~~

- ☒ ~~Figure out where to get the 3 cameras from (images doesn't have to be video)~~
- ☒ ~~Establish what development environment is dependent on (ie. in lab or home setup)~~
- ☒ ~~Run, save, and load extrinsic and intrinsic camera calibration for a set of three cameras~~
- ☒ ~~Capture images and videos from up to three cameras simultaneously~~

Computer Vision (3D Reconstruction)

- ☒ ~~Create virtual environment with pycharm (transition from jupyter notebook or google collab)~~
- ☒ ~~Explore OpenCV single camera calibration [documentation](#)~~
- ☒ ~~Explore OpenCV multi-camera calibration package~~
- ☒ ~~Include epipolar geometry~~
- ☒ ~~Apply opencv image distortion reduction functions~~
- ☒ ~~Calculate positions on image planes~~
- ☒ ~~Extract a depth/disparity map from the images~~
- ☒ ~~Save camera calibration parameters to external numpy file~~
- ☒ ~~Implement 3D stereo mapping [tutorial](#) (extend to 3 cameras)~~

New Approach

- ☐ [Calculate](#) the camera projection matrix
 - ☐ Formula for converting world coordinates to pixel coordinates
- ☐ Stereo [rectify](#) should give an estimate of the rotation and translation matrix
- ☐ Use the stereo triangulation [function](#) to calculate the 3D coordinates from 2D image points

Moral of the story : we don't need an intense depth map or 3d reconstruction of the entire workspace. StereoBGM is only useful when

we want to do this. But for our problem, we **already know** some points
e.g. **aruco corners** or tdcr disks.

we can **simply pass** them **to cv2. triangulate**.

The **catch** is that cv2.triangulate **requires** you **to send** the **same object coordinate observed in both images** (ie. one corner of the aruco marker).

So epilines can come into play helping you search for the corresponding images.

So basically everything you've implemented now comes together to give you 3d world coordinates.

Documentation (Ongoing)

- ☒ ~~Include main docstrings for methods~~
- ☒ ~~Potentially add unit test cases/testing suite for calibration phase or depth mapping~~
- ☒ ~~Include in code comments~~
- ☒ ~~Collect cool image and video results to keep track~~
- ☒ ~~Compile code into one big file or put into OOP~~
- ☒ ~~[Report](#) double column format (preferable LaTeX): introduction/story shape sensing, state-of-the-art, methodology, project timeline, results, future work, conclusion~~
- ☒ ~~Complete report on [LaTeX](#)~~
- ☒ ~~Start the research poster~~
- ☒ ~~Work on intermediate presentation~~

Potential Project Extensions

- ☒ ~~Conduct a TDGR robot trial~~
- ☒ ~~Implement a visualizer GUI with QT~~

- ☒ ~~Determining the pose of the continuum robot's joint to tip and shape mapping with machine learning~~
- ☒ ~~Image segmentation of TDGR features (e.g. spacer disks)~~
- ☒ ~~**Shape Sensing and extract disc location: put ArUco Markers onto TDGR's discs and use Aurora System for the ground truth**~~
- ☒ ~~Could help Priyanka & Hanna with 3D Euler Curves w/ ML (could be fall project) get shape of backbone and calculate shape~~

Good debugging source:

- [Tim Poulsen – Acquiring images with OpenCV](#)
- Made open source github contributions for my issue solution: [here](#) and [here](#)

June 14th, 2021.

Implementation Tasks: Pose Estimation Phase

Key Concepts

One Proposed Approach for Aruco Marker World Coordinates

Goal: to get the x/y/z coordinates at the center of the ArUco marker, and the angle in relation to the calibrated camera.

- X and Y coordinates in aruco can be determined by the average of the corners

- The angle relative to the camera can be determined by the Rodrigues of the rotation vector, the matrix must be filled prior
- Finally, yaw pitch and roll can be obtained by taking the RQ Decompose of the rotation matrix

June 17th, 2021.

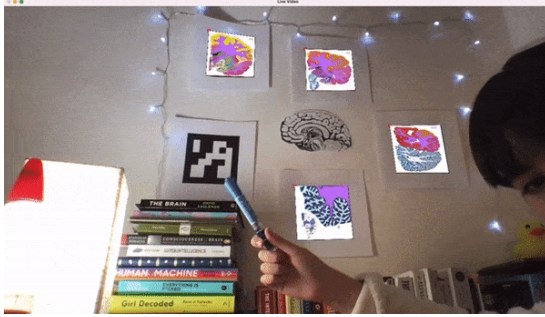
Weekly Supervisor Check In

Complete by June 24th.

- Implementing **camera calibration** for an available camera (print out a chessboard- add cardboard to background)
- Manual on chessboard movements
- Checkpoints
- Intrinsic and Extrinsic Parameters work
 - Transformation matrices for stereo matching
 - Depth matching
 - Establish ground truths (ie. rubik's cube with known dimensions and checked dimensions with multicam system-calibration)
- Relative positions of the cameras (how multi cam)
- Future steps: create checkpoints to meet for the multicam system
- Ask for data or collect 3 angles with cameras

Completed

- Augmented Reality ArUco detection for neuroanatomy and released [here](#)



- Live video ArUco marker detection

June 23rd, 2021.

Weekly Lab Meeting + Dr. Burgner-Kahrs Office Hour

Key Takeaways

- A high-level overview of the field of robotics and what it looks like to contribute to it (vs other fields), etc.
- Neurosurgery field
 - Really intensive work (140 hour work weeks)
 - Patient death trauma
 - Depression, stress, substance abuse
- Establish a meaningful purpose of contact
- Be a future grad student in their lab? (Contact prof and post docs for example, ask for context)
- Problem solving a specific project the prof built (could contact grad student instead of prof, the 1st authors instead of the last author which is usually the prof)
- Really challenging to catch up with hundreds of emails per day
- A Master's degree is looking intriguing!

Master's Degree

- French system: master's degree is 4th year b/c undergrad is 3 years
- Self-taught school
- 2 years
- Research-dependent and main project
- Deadlines
- Grades emphasis
- AI research
- Professors validating tests
- First half is specific courses and a project
- Use a lot of math
- Depends on context for undergraduate requirement
- Barbara Mazzolai
- BCI Master's
- Sorbonne University AI [Master](#)
- A survival guide to a [PhD](#)
- Cognitive Science [Master Program](#) in Paris
- Master [programs](#) in BCIs (Brain-Computer Interface)
- Braini BCI [Lab](#) (MIT Media Lab)

[Mehdi Khamassi](#)

[Adam Marblestone](#)

<https://www.lesswrong.com/posts/jrewt3rLFiKWkuyZ/big-picture-of-phasic-dopamine?commentId=WK7kEMypn6a423rac>

[Steve Byrnes's Homepage](#)

[Big picture of phasic dopamine](#)

[Careers](#)

[Kernel logo](#)

[About Ariel](#)

[People details](#)

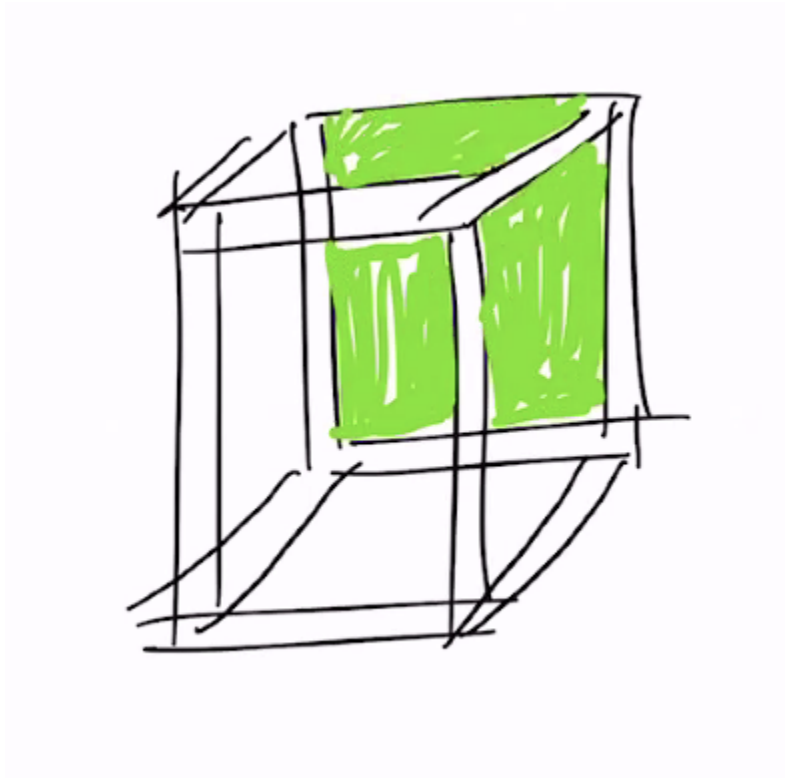
[Smarti Gras: UTM's summer celebration of undergraduate research | Research](#)

Complete by June 24th.

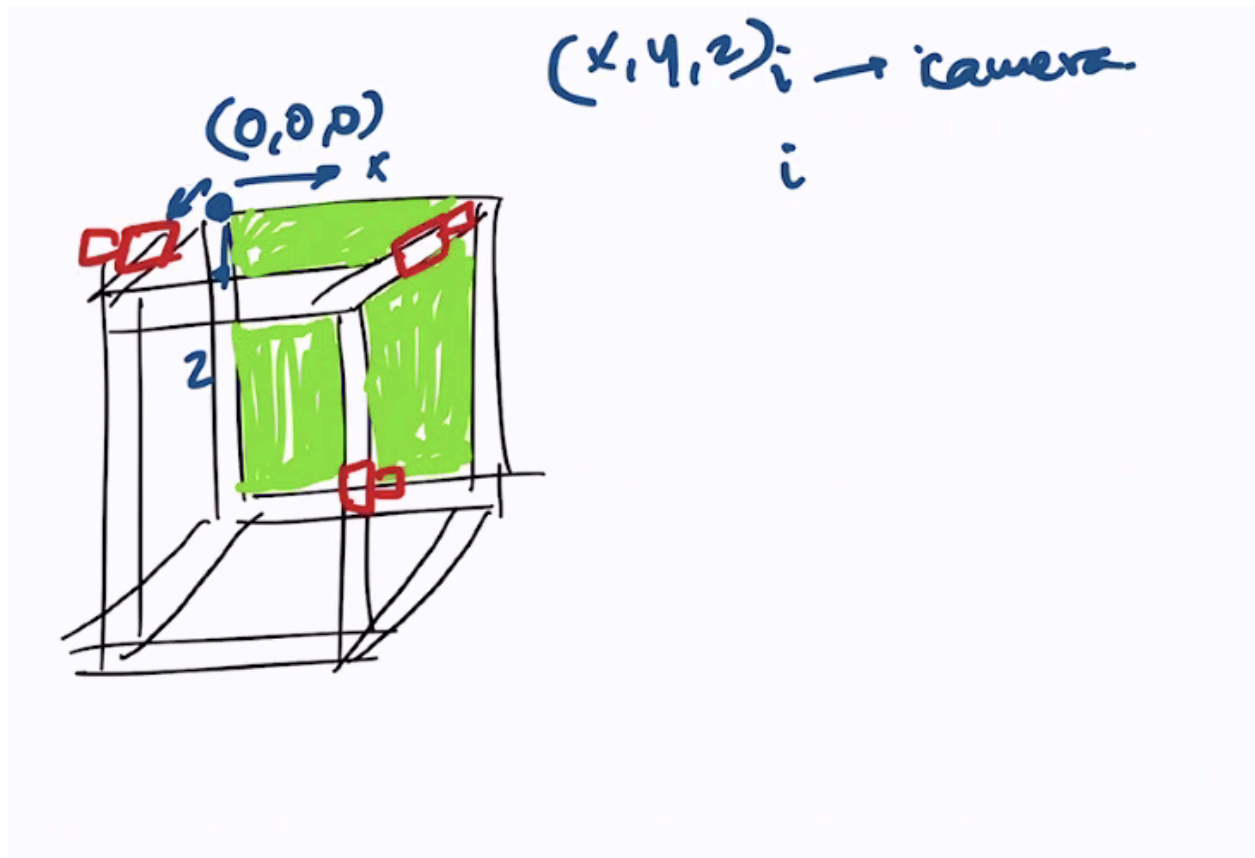
- Collect images with Quentin from lab
 - Coordinate schedule
 - Send file setup system- calibration code
 - Try running code on system
 - Documented code
 - Send chessboard image
 - Get chessboard cardboard physically
 - Maximum: 30 images (?)
 - Google good amount required for calibration
 - Accurate coordinates of the 3 cameras (as best as possible)
 - Select one corner of rail and measure x,y,z coordinates
 - Stick another cardboard
 - Priyanka: 9am-12pm EDT works best

- Training Process
 - Extension for ground truth: 30 images

Green Screen

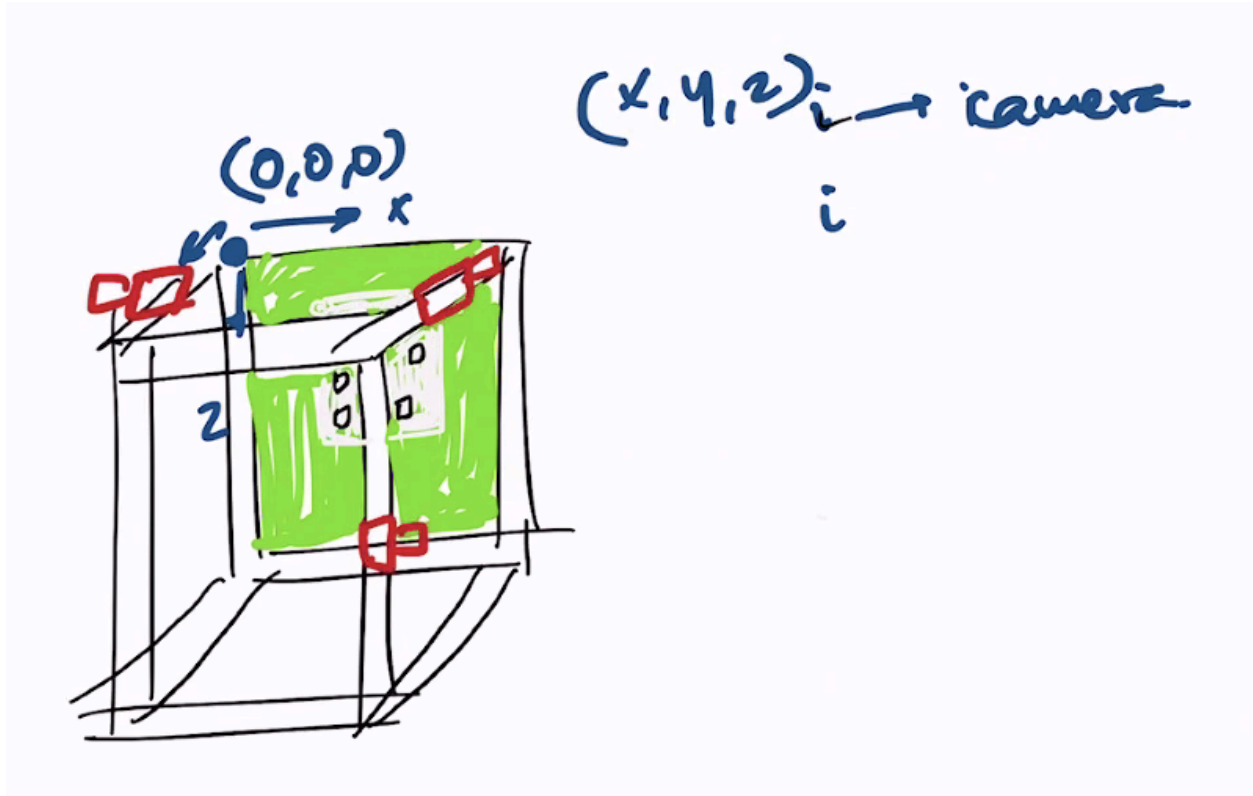


Pick one default corner as the origin and find the x,y,z coords of each camera



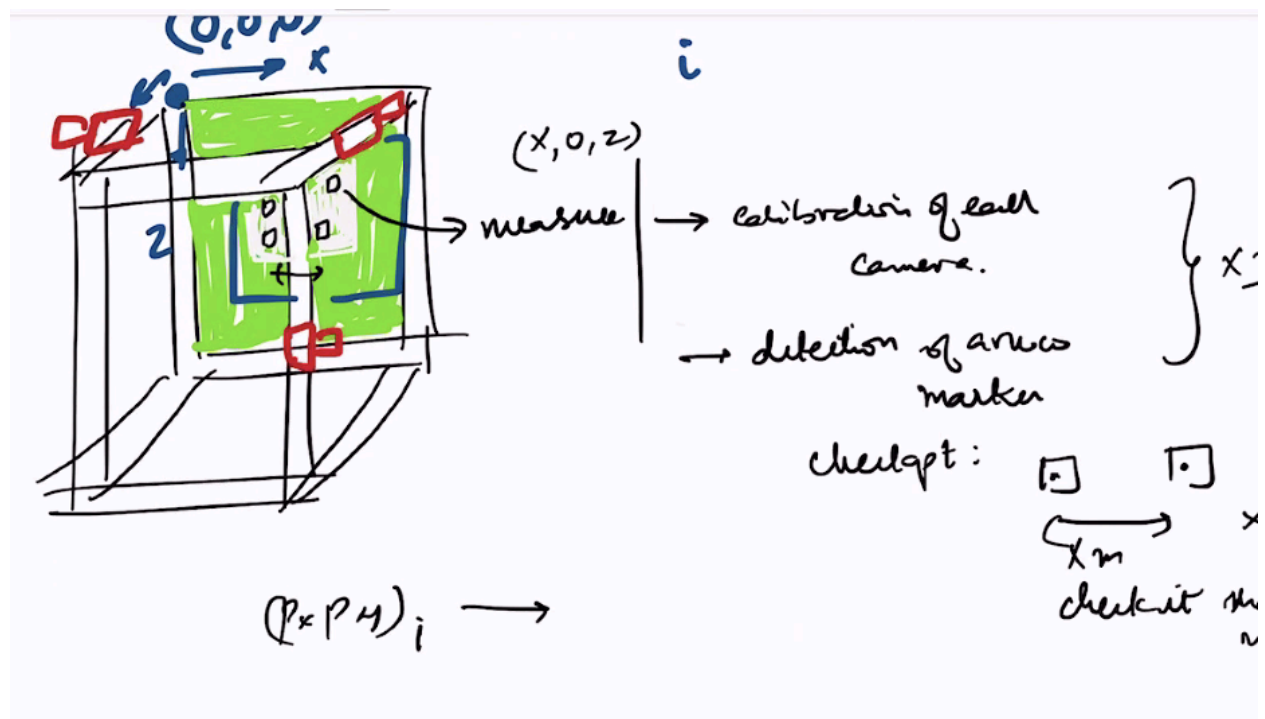
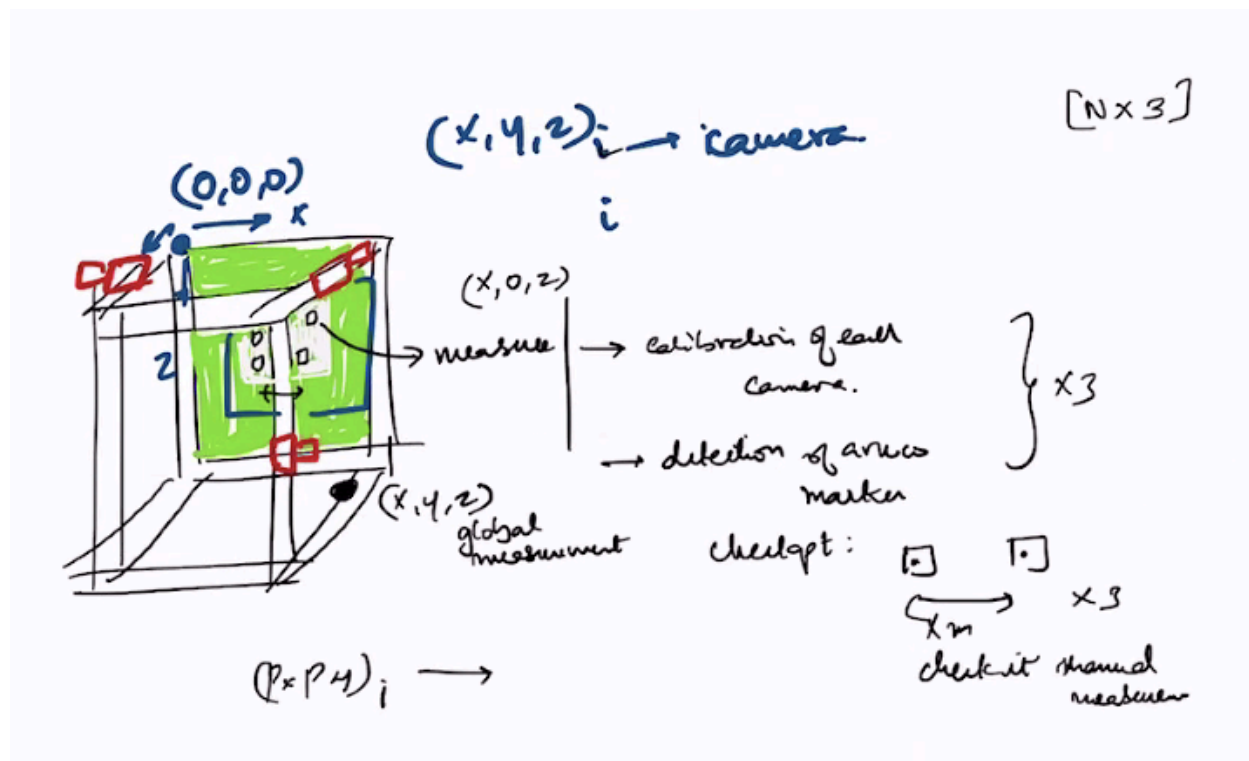
Know which camera is which and find the i correspondence of each camera because $[N \times 3]$ images will be outputted and need to know which images are for 3 cameras

Get sheet with aruco markers (should help with ground truth, despite depth matching) and add to end



Doing here:

1. Calibration of each image
 2. Detection of aruco markers module
 3. Doing 3 times ^
 4. Checkpoint: calculate the distance between aruco markers in cm (measure with ruler first and check distance if the code and camera outputs)
 - a. Input: pixel position of each aruco marker
 - b. Output: what
- Need a **reference point** to get a global frame of reference
 - Measure distance of each aruco marker from a checkpoint with x,y,z coords (with a ruler?)
 - A point where all 3 cameras should be able to see



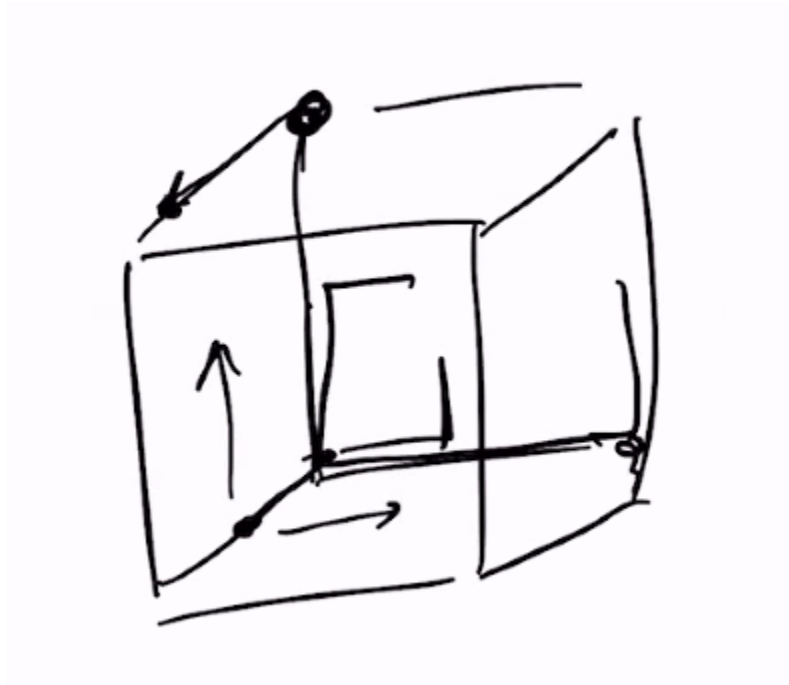
- All measurements with respect to global origin

Two image sets

- Calibration 3XN
- Reference (put colourful marker object- thing looks like a pacifier)

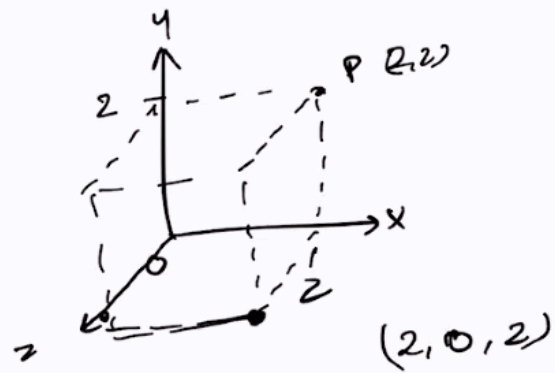
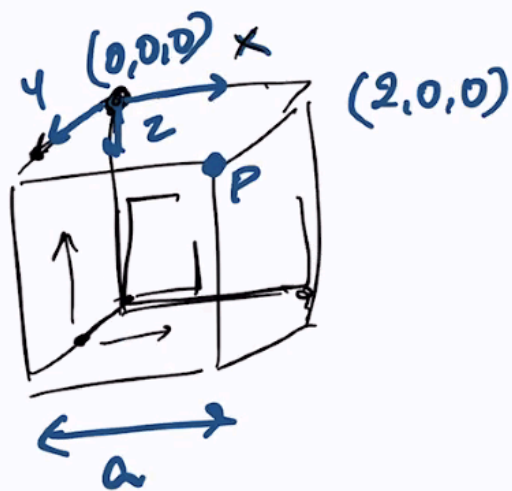
Calibration 3XN	Reference
Chessboard	stereo marker Reference

-

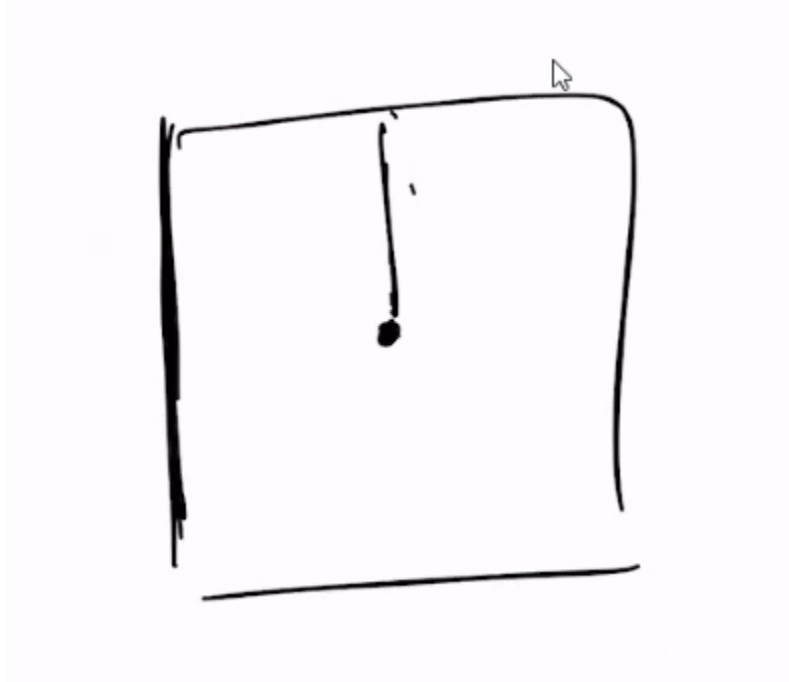


- **NEW IDEA:** choose a global point that can be seen by all 3 cameras, that will also act as the reference point

How to calculate x,y,z with a ruler



If global coord/reference point is middle, make sure it's really centered



- Run calibration
- Run aruco first
- first , calibrate cameras with chessboard and move it around manually (calibrates camera)
- Aruco marker stuff (fix aruco marker into green)
- Next items: read up on stereo matching
- Meet up with priyanka beforehand if have doubts before real data collection
- Look at existing implementations of multi camera calibration
- Figure out how it works
- What parameters we need (refer to code and our drawings)

- Get images from Quentin
- Compile all functions into one big file
- No video feed
- Have classes for each module in compiled code file
- Two sets of images run uniquely each for module
 - Exact environment
 - Don't need chessboard for aruco marker

Basic:

- Have aruco marker detected
- Could have pose estimation for markers

Basically:

- Apply code to new images
- Get image dataset
- Compile file into one
- Text priyanka if done early

June 28th, 2021.

Optimizing Camera Calibration Scores + Stereo Matching Research

Main Subtasks

- **Some quick links for stereo mapping:**
 - Stereo matching (two cameras) with code: [Stereo 3D reconstruction with OpenCV using an iPhone camera. Part III.](#)
 - Calibration using multi cameras: [Easy-to-Use Calibration of Multiple-Camera Setups](#)

- Test it over the next couple of days by simply copy pasting and seeing what it does
- Revisit extra opencv features: [OpenCV: ArUco Marker Detection](#)

Stereo 3D reconstruction requires the following steps:

- **Camera calibration:** Use a bunch of images to infer the focal length and optical centers of your camera
- **Undistort images:** Get rid of lens distortion in the pictures used for reconstruction
- **Feature matching:** Look for similar features between both pictures and build a depth map
- **Reproject points:** Use depth map to reproject pixels into 3D space.
- **Build point cloud:** Generate a new file that contains points in 3D space for visualization.
- Build mesh to get an actual 3D model (extra)

1. Analyze every image frame of a video feed as the user moves the pattern in front of the camera until the pattern is detected a number of times (this is normally an arbitrary measurement but the general rule is to have around 10 detections at least).
2. Get a chessboard pattern.
3. Put a chessboard pattern on a white wall (ideally).
4. Take several pictures of the pattern from different angles where the chessboard moves, not the camera (minimum: 10 photos, take a lot to have variety to pick which best calibrates).
5. Output camera parameters from code: camera matrix (K), distortion coefficients (dist), and the rotation and translation vectors (rvecs and tvecs).
6. Then could simply save those camera parameters into a numpy file and load it later. Why numpy and not XML or JSON? because with

numpy files there's no need to parse the data. Or manually save the runs on a notes file.

7. To do 3D reconstruction, we need 3 parameters: the camera matrix, the distortion coefficients and the focal length. The focal length can be derived from the camera matrix.

Parts I & II complete, then do:

[Stereo 3D reconstruction with OpenCV using an iPhone camera. Part III.](#)

Extend mechanical obstacle avoidance model 2D mapping to 3D using machine learning:

- ML learn difference in deviations
- Measure tip pose and each disk's position
- Could have aruco

Ideas to lower reprojection calibration error:

- Chessboard on the left or right could help better error distortion detection
- Entire chessboard should be visible

July 3rd, 2021.

Investigating 3D Reconstruction

Stereo Mapping Research Material

- Extract a depth/disparity map from the images
- Save camera calibration parameters to external numpy file
- Implement 3D stereo mapping tutorial (extend to 3 cameras)
 - Can't find material for 3 cameras, only 2

Stereo Mapping Research Material

- Stereovision parameter tuning + GUI link to adjust: [Calculating a depth map from a stereo camera with OpenCV](#)
- https://web.archive.org/web/20150312053300/http://docs.opencv.org:80/trunk/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html
- [erget/StereoVision: Library and utilities for 3d reconstruction from stereo cameras.](#)
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html#py-depthmap
- [OpenCV: Depth Map from Stereo Images](#)
- [3DReconstruction/disparity.py at master · OmarPadierna/3DReconstruction · GitHub](#)
- [Stereo 3D reconstruction with OpenCV using an iPhone camera. Part III.](#)

July 7th, 2021.

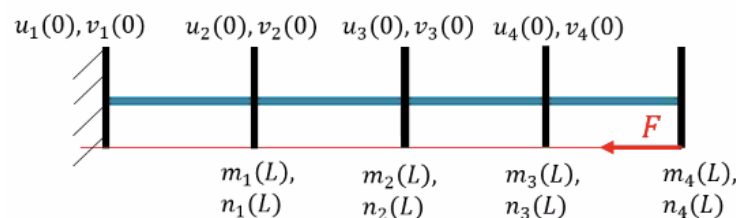
Weekly Lab Meeting

Quentin's Research Project Presentation

- Load bearing vs compliance of robots
- Buckling and stiffness variation
 - Stiffening: control for keeping the backbone straight and increase critical force
- **Contributions**
 - Study stability issues specific to TDCR
 - Propose the use of buckling for stiffness modulation
- **Approach**
 - Generic forward and inverse models of planar TDCR
 - Numeral analysis based on continuation methods and bifurcation analysis
 - Experimental validation

Methodology:

- **Model**
 - VCref model from the Frontiers paper (tendon forces and friction at each desk)
 - States for Planar + Kirchhoff rod
 - Solved with shooting method



- **Computing the robot shape**
 - Challenges: large variation of robot shape to elastic instabilities (difficult to predict), problem of convergence for numerical solvers
 - Continuation method: prediction tangent and newton's raphson correction process

- Detection of branching points
- **Experimental setup**
 - Microscribe tactile scanner
 - Planar TDCR 1-2 disks
 - Manual actuation calibrated weights
 - Parallel NiTi rods (prevents contact between tendons and backbone)
- **Results**
 - Model validation
 - Stability analysis (n=1)
 - Stability analysis (n)
 - Stiffness modulation

July 10th, 2021.

Weekly Supervisor Check In

Debugging

- Implementing the GUI for the disparity map to get better results
- Capture images from phone to get the extractable EXIF files
- Got the FLIR camera specs manual to confirm focal length, pixel size, etc. (not sufficient data though)
- [Camera Calibration Toolbox for Matlab](#)
- Properties of Intrinsic Camera Parameters Matrix: [What Is Camera Calibration? - MATLAB & Simulink](#) & [Dissecting the Camera Matrix, Part 3](#)

- The intrinsic parameters include the **focal length**, the optical center, also known as the principal point, and the skew coefficient. The camera intrinsic matrix, K, is defined as:

$$\begin{bmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

○

- Disparity Map Parameter Tuning GUI: [Building an interactive GUI with OpenCV | Stackable](#)
- Novel [approach](#) to 3D real world coordinates from two perpendicular 2D images
- Manual calculation of camera's focal length (b/c lab cameras don't have this metric or an EXIF): [Focal length from calibration parameters - OpenCV Q&A Forum](#)
 - Priyanka got 8.1mm with manual calculator
 - Yasmeen predicted 18mm for the FLIR camera's focal length and store catalogue history
 - Camera model: FLIR BLACKFLY S (P/N BFS-U3-51S5)
- [Approximate Focal Length for Webcams and Cell Phone Cameras](#)
- [How to calculate the focal length in mm by the result of camera calibration?](#)

July 12th, 2021.

Shape Sensing Processing Pipeline

Steps

- **Keyframe selection:**
 - **Requirements:**
 - Baseline distance in 3D between both frames: the larger, the better – but at least 4 cm.
 - Area overlap: depth can only be directly computed for areas that are visible in both images. Thus, the minimum is 40% overlap; but again: the larger, the better.
 - Errors: if the 6DoF has low confidence in its motion estimate, the image should not be chosen. Must select captured images that are distinct with relative 6 degrees of freedom (6DoF) movement.
- **Preprocess** camera images to prepare them for matching.
 - **Camera calibration** to infer the focal length and optical centers of your camera
 - **Keypoint detection**
 - using the traditional SIFT algorithm
 - **2D Keypoint matching** between both images.
 - need the best key points where we are sure they are matched in both images to calculate reprojection matrices.
 - detect which key points are present in both images, as well as their difference in the position.
 - Using the FLANN (Fast Library for Approximate Nearest Neighbors) matcher algorithm to pick the best potential matches between similar key points in both frames based on their distance using a K-nearest-neighbor search.
 - Keypoint matches are drawn
 - **Stereo rectification with Epipolar Lines:** undistort and align images to each other.

- Using the reprojection matrices, we can rectify the images to a common image plane. Matching keypoints are on the same horizontal epipolar line in both images.
 - This enables efficient pixel / block comparison to calculate the disparity map (= how much offset the same block has between both images) for all regions of the image (not just the key points!).
- **Stereo matching** between these selected images to create a disparity / depth map.
 - Use a depth map to reproject pixels into 3D space.
 - Build point cloud: Generate a new file that contains points in 3D space for visualization.

Independent research + potentially recommend 2 camera system but test out lab pics first (just say “stuck” for now in prez in road map for extra like 3 lab cameras post processing/extras point)

Additional to do's:

Optimized setup with 2 cameras over 3?

Convert pixel data to 3d coords from disparity map (calculate the x, y, z equation or point cloud mesh lab

Why 2 cameras:

Produces more accurate disparity map(personal experience, 2 cameras was better than the lab photos)

Good timeframe for 2 cameras given lack of existing open source code for 3 cameras

So compensate in other ways that would've spent on setting up 3 cameras
like keyframe requirements are optimized like DOF

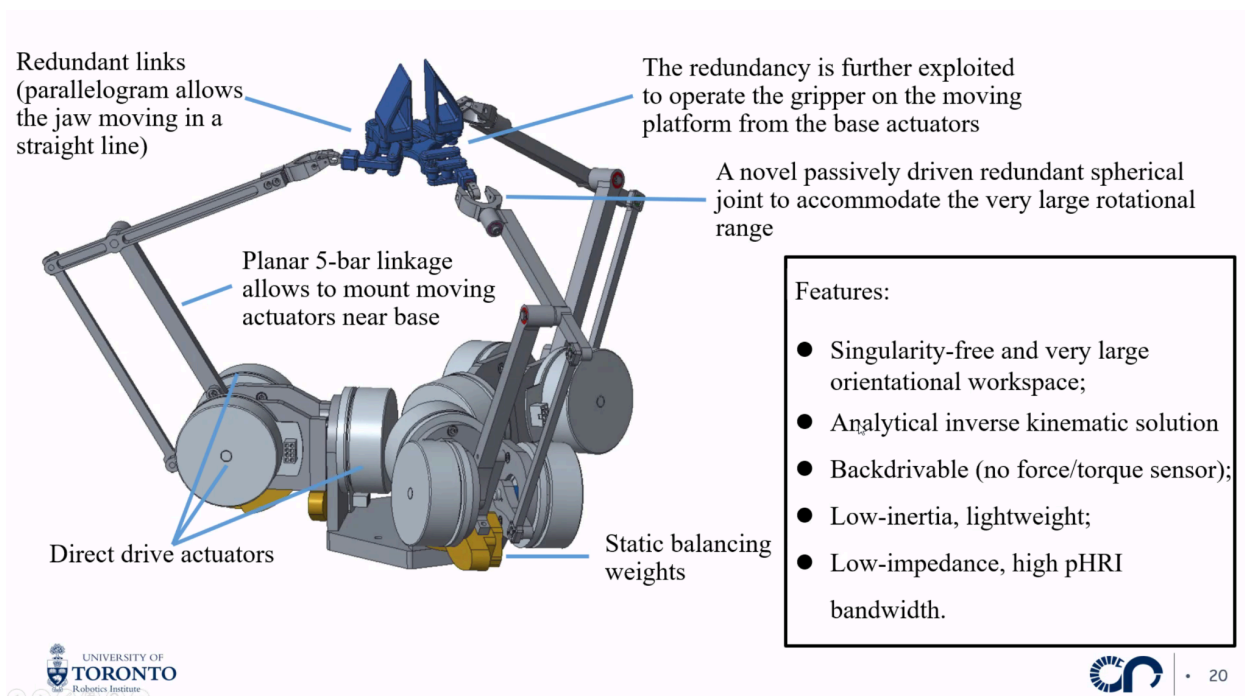
Connect to TDCR extension

“Let's do joint research” and then meet up with jessica

July 21st, 2021.

Parallel Continuum Robots Exposé

Key Takeaway Components



Priyanka's Navigation Explanation (future directions inspiration)

8:54 PM Wed Jun 16

PHD

Home Insert Draw View

A Text Mode

Lasso Select

Insert Space

We couldn't merge some changes. To review conflicts, use OneNote on your computer or OneNote Online.

< Summer 2021 Edit

Catheter modeling

Objectives : Tackle acute isc...

ICRA2021

A Translational Parall...

EC

No additional text

MCR project

No additional text

Thesis

Look st papers with c...

Stereo camera ROP

May : Weeks 1-2 : OpenCv P...

Recording

Use Transcription (Closed Captioning) has been enabled. Who can see this transcript?

View Options

2. Visual Autonomous Navigation Controller

- Obstacle heat map: characterizing obstacle distribution
 - Gather a point cloud of obstacles using a depth camera
 - Filter the points and combine into discretized cells for an $n \times n$ heat map
 - Calculate the intensity of each cell and classify cells as either *blocked*, *open*, or *occluded*
 - Determine the desired locomotive direction by minimizing the blocked cells on the way

$$\epsilon = c \cdot \left(\max_{x,y} I_{x,y} + \text{median}(I) \right)$$

$$B(x,y) = \begin{cases} 1 & \text{if cell } (x,y) \text{ is blocked} \\ 0 & \text{if cell } (x,y) \text{ is occluded} \\ -1 & \text{if cell } (x,y) \text{ is open} \end{cases}$$

A

B

blocked

open

occluded

© Copyright National University of Singapore. All Rights Reserved.

Audio Settings

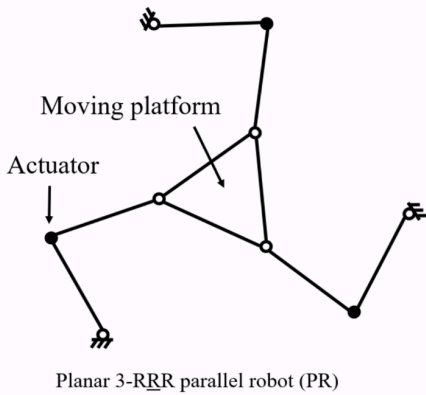
Rate: Normal

Live Transcript

Leave

+ Page

III. Kinematic analysis of tendon-driven parallel continuum robots

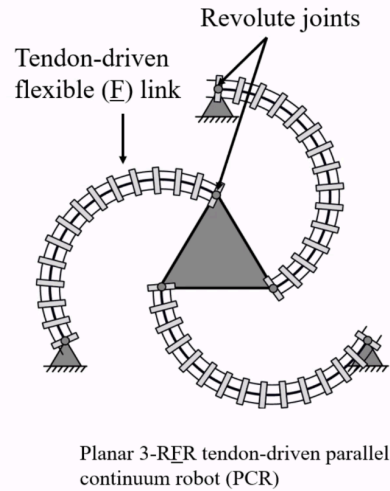


Compared with their serial counterparts

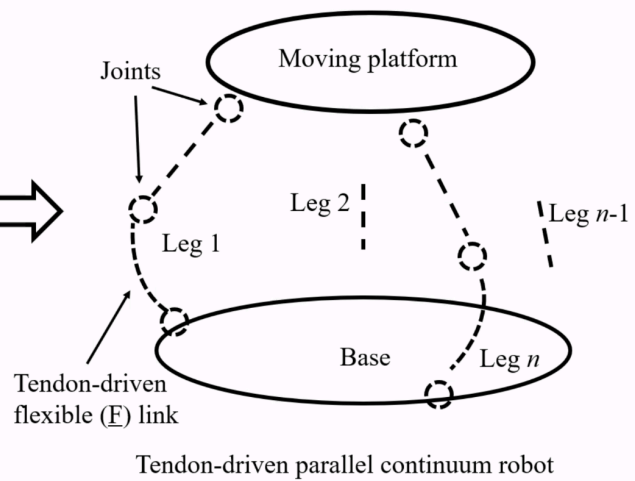
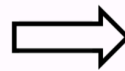
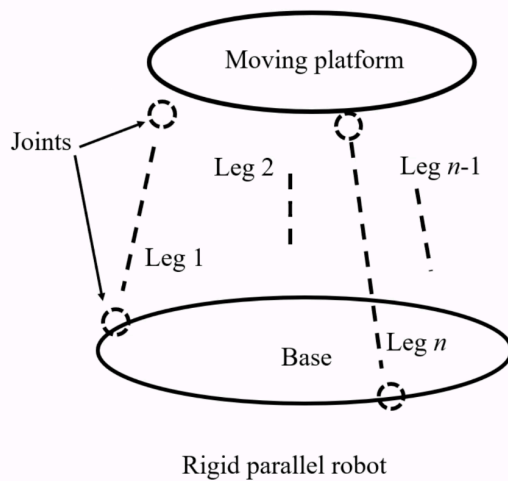
- High stiffness
- High accuracy
- Large load capability
- Faster

PCR vs. PR

- Inherent compliance (human-safe)



Tendon-driven parallel continuum robots



Velocity equations

Constraint equations and time derivative

$$(\mathbf{p} + \mathbf{Q}\mathbf{b}_i - \mathbf{a}_i)^T (\mathbf{p} + \mathbf{Q}\mathbf{b}_i - \mathbf{a}_i) = \rho_i^2, \quad i=1,2,3$$

$$(\mathbf{p} + \mathbf{Q}\mathbf{b}_i - \mathbf{a}_i)^T (\dot{\mathbf{p}} + \dot{\mathbf{Q}}\mathbf{b}_i) = \rho_i \dot{\rho}_i$$

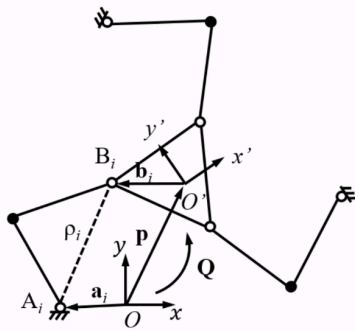
ρ_i and κ_i of the PCR:

$$\rho_i = \frac{2}{\kappa_i} \sin\left(\frac{l_i \kappa_i}{2}\right)$$

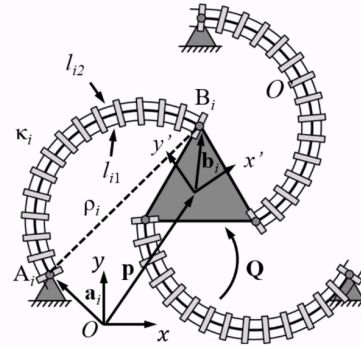
$$\kappa_i = \left| \frac{l_{i2} l_{i1}}{d(l_{i1} + l_{i2})} \right|$$

l_i : length of the backbone

d : distance between tendon and backbone



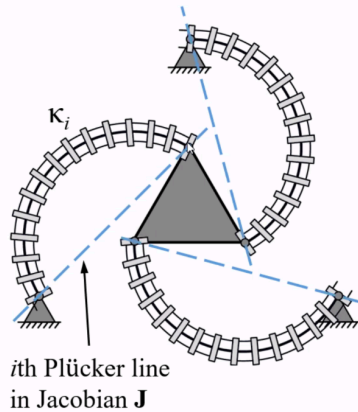
Velocity equations: $\mathbf{J}\mathbf{t} = \mathbf{K}_r \dot{\mathbf{p}}$
Static equations: $\boldsymbol{\tau} = (\mathbf{J}^{-1} \mathbf{K}_r)^T \mathbf{F}$



Velocity equations: $\mathbf{J}\mathbf{t} = \mathbf{K}_c \dot{\mathbf{l}}$
Static equations: Ongoing work

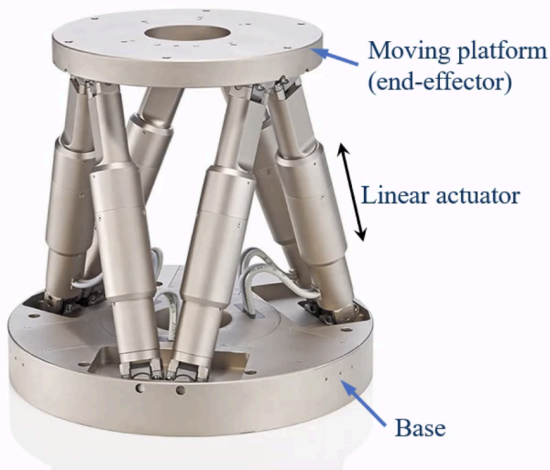
Singularity conditions

Velocity equations: $\mathbf{J}\mathbf{t} = \mathbf{K}_c\dot{\mathbf{l}}$



Type I (serial) singularities (Jacobian \mathbf{K}_c becomes singular)	Type II (parallel) singularities (Jacobian \mathbf{J} becomes singular)	
$\begin{cases} \kappa_i = 0, \text{ fully extended} \\ \kappa_i = \kappa_{imax}, \text{ fully folded} \end{cases}$	<p>Intersect at a common point</p>	<p>Parallel to each other</p>

I. Parallel robots and applications



Gough-Stewart platform (Tempel et al., 7th IFToMM International Workshop on Computational Kinematics 2017)

What is a parallel robot?

A generalized parallel manipulator is a **closed-loop** kinematic chain mechanism whose end-effector is linked to the base by several **independent** kinematic chains.

--- J.-P. Merlet, "Parallel Robots", Springer Netherlands, 2nd edition, 2006

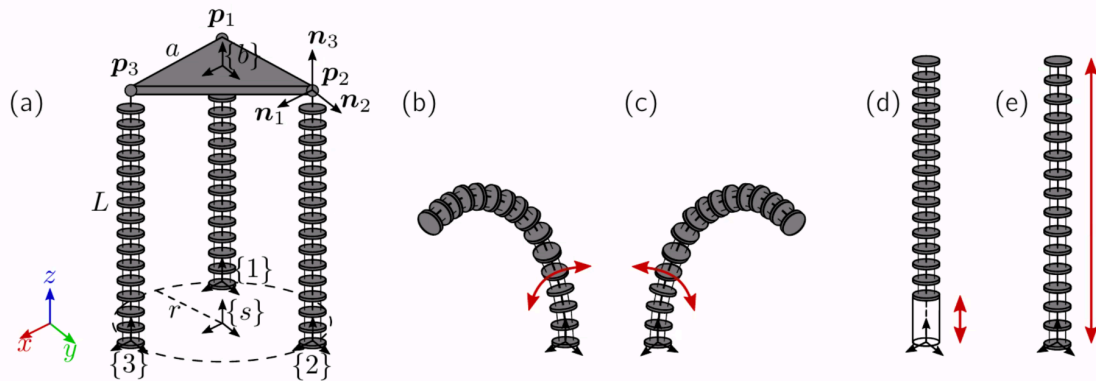
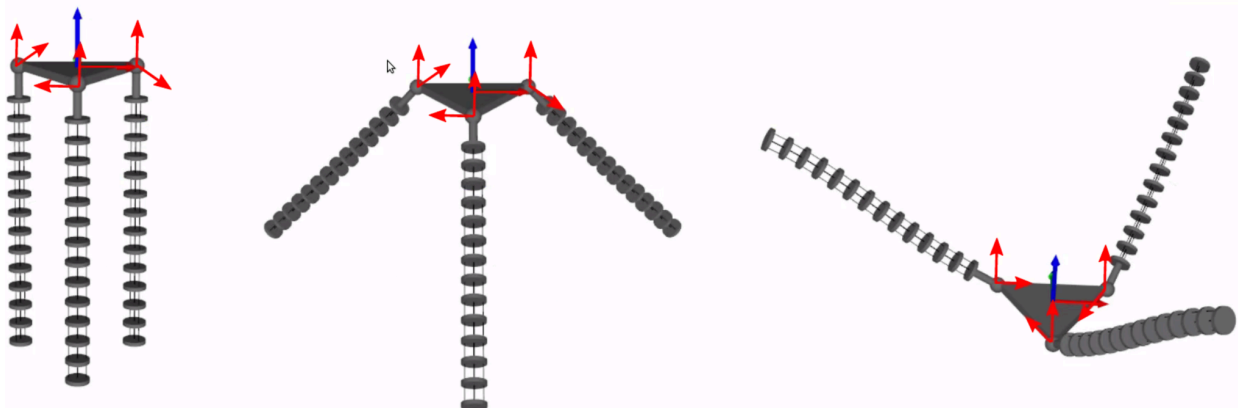


Figure: (a) General common structure of the proposed parallel continuum robot designs. (b-e) Different ways to actuate each of the tendon actuated continuum robots: (b) Bending in x -direction of the local base frame; (c) Bending in y -direction of the local base frame; (d) Translating the base of the robot in z -direction of the local base frame; (e) Elongation of the robot structure.

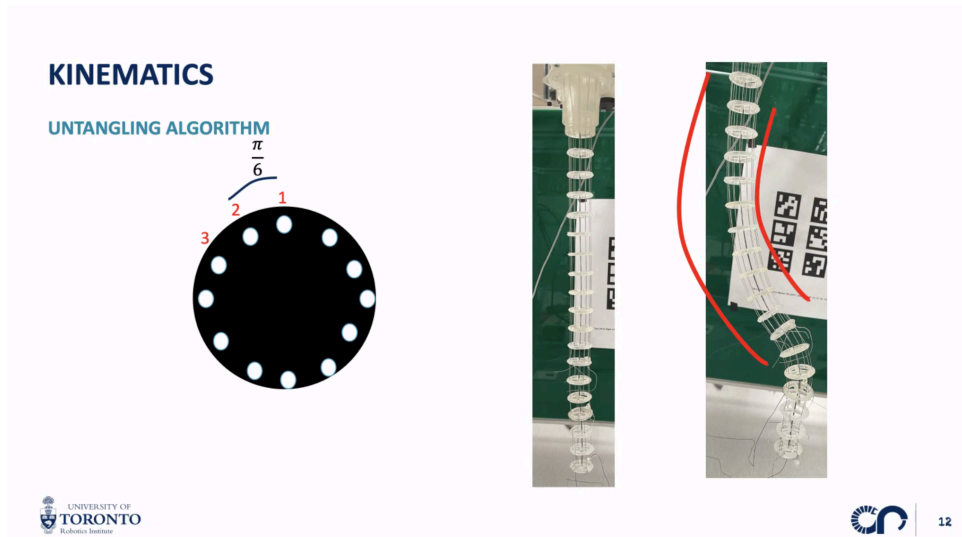


July 28th, 2021.

Mathew's Shape Sensing Research Presentation

Key Takeaway Components

- **Untangling Algorithm:**

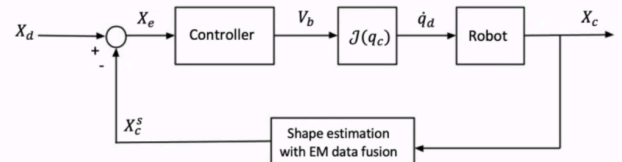


- **Closed Loop Control Mechanism:**

CONTROL

CLOSED LOOP CONTROL

- Task space control
 - Jacobian based control using a finite differences approach
1. Evaluating forward kinematics at q_c
 2. applying a small step to the i th component of q_c
 3. Evaluating Forward kinematics at q_c^*
 4. Computing the difference $FK(q_c) - FK(q_c^*)$
 5. Dividing the difference by the small step for the i th column of the jacobian



- X_d : Desired end effector configuration
- X_c : Current end effector configuration
- X_e : Error in task space
- V_b : Task space end effector velocity
- \dot{q}_d : Joint space velocity d
- X_c^s : Estimated state from fusing the EM sensor data and the current configuration from the kinematic mode



14

- **Electromagnetic Sensor Integration**

- Can't have an EM sensor along each disc (system has limit of 4 sensors)

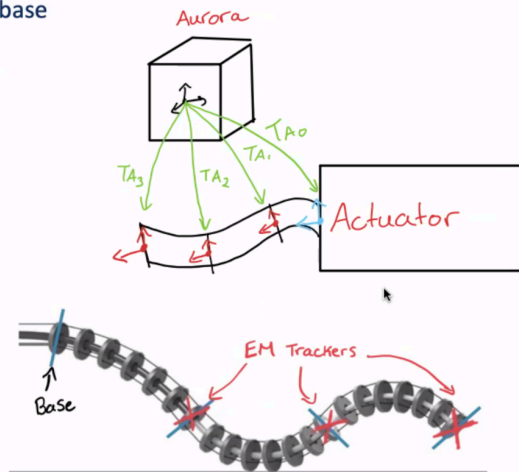
EM SENSOR INTEGRATION

- Making everything relative to the actuator base
- Aurora gives the transformation matrix per sensor relative to the emitter
- T_{A0} = origin point (actuator base)

$$T_{01} = T_{A0}^{-1} * T_{A1}$$

$$T_{02} = T_{A0}^{-1} * T_{A2}$$

$$T_{03} = T_{A0}^{-1} * T_{A3}$$



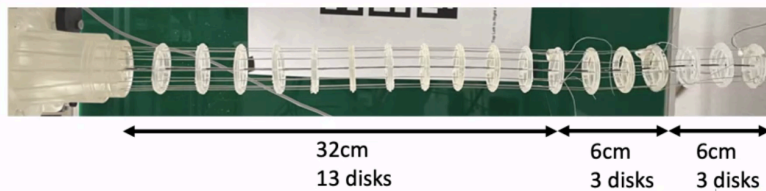
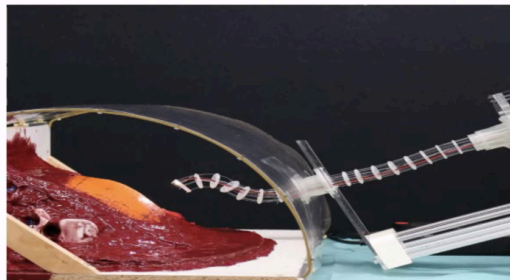
MEDUSA

STRUCTURE

- Tendon driven continuum robot
- 3 segments and 4 tendons
- Antagonistic tendons

APPLICATION

- Minimally invasive surgery
- Laparo-endoscopic single site surgery
 - Performed by small incisions in the abdomen
- Open ports for tools



Aug 4th, 2021.

Quentin's Research Presentation

CONTINUUM ROBOTS

- Inspired by biological appendages
- Dexterous and flexible
- Continuously bending and flexible backbone
- Exhibit the capability of exploring confined spaces
- Can range in size
- Applications:
 - Medical applications
 - Search and rescue
 - Industrial



CHALLENGES FOR MINIMALLY INVASIVE SURGERIES

CHALLENGES

- Shape sensing
- The body of the robot can collide with internal walls or organs
- External disturbances can disrupt the shape of the robot

APPROACHES

- Model based shape reconstruction
 - Relies on the continuum robot's kinematics
 - Complexity, accuracy, and computational expense
- Sensor integration
 - Sensor data can be noisy
 - Cannot obtain the entire shape of the robot on its own

RESULTS

- Compare the position of each disk from the EKF to the ground truth
- The ground truth is the position of each disk measured using the faro arm
- Compare the results from the ekf vs using only the CC model in different configurations
- Apply a force at the zero configuration and measure the difference

CURRENT PROGRESS

- Robot setup: Done
- EM integration: 90% (fixing an issue recently found)
- EKF algorithm : Done
- Get results: 50%
 - Getting ground truth: Done
 - Collect data: TODO
- Paper write up: 0% (starting Monday)

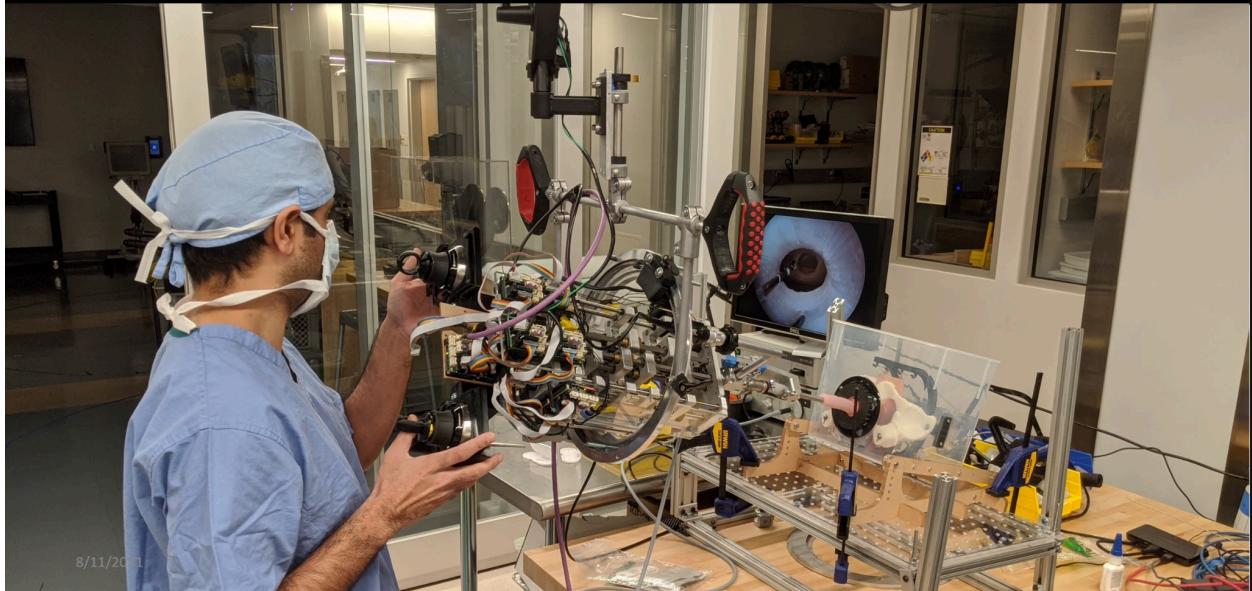
PAPER BREAKDOWN

- Introduction
- Materials and Methods
 - Experimental setup
 - Sensor integration
 - EKF algorithm
- Evaluation and Results
 - How ground truth is obtained
 - Showcasing results
 - Comparison to CC model
 - (Possibly simulation)
- Conclusion

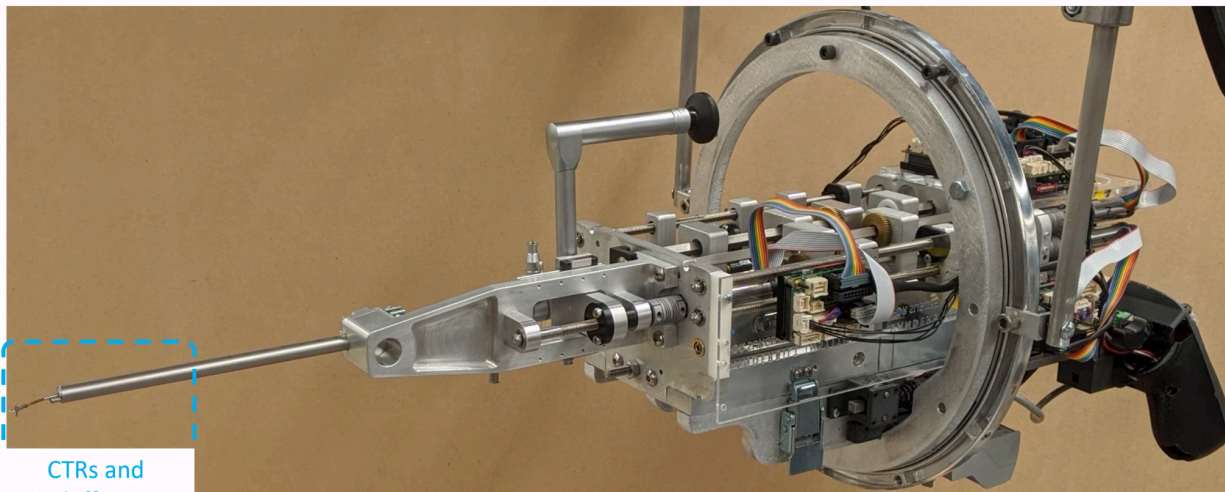
Aug 11th, 2021.

***Guest Speaker, Dr. Ernar Amanov
(former PhD student of Jessica and
postdoc at Vanderbilt University)***

Overall Setup

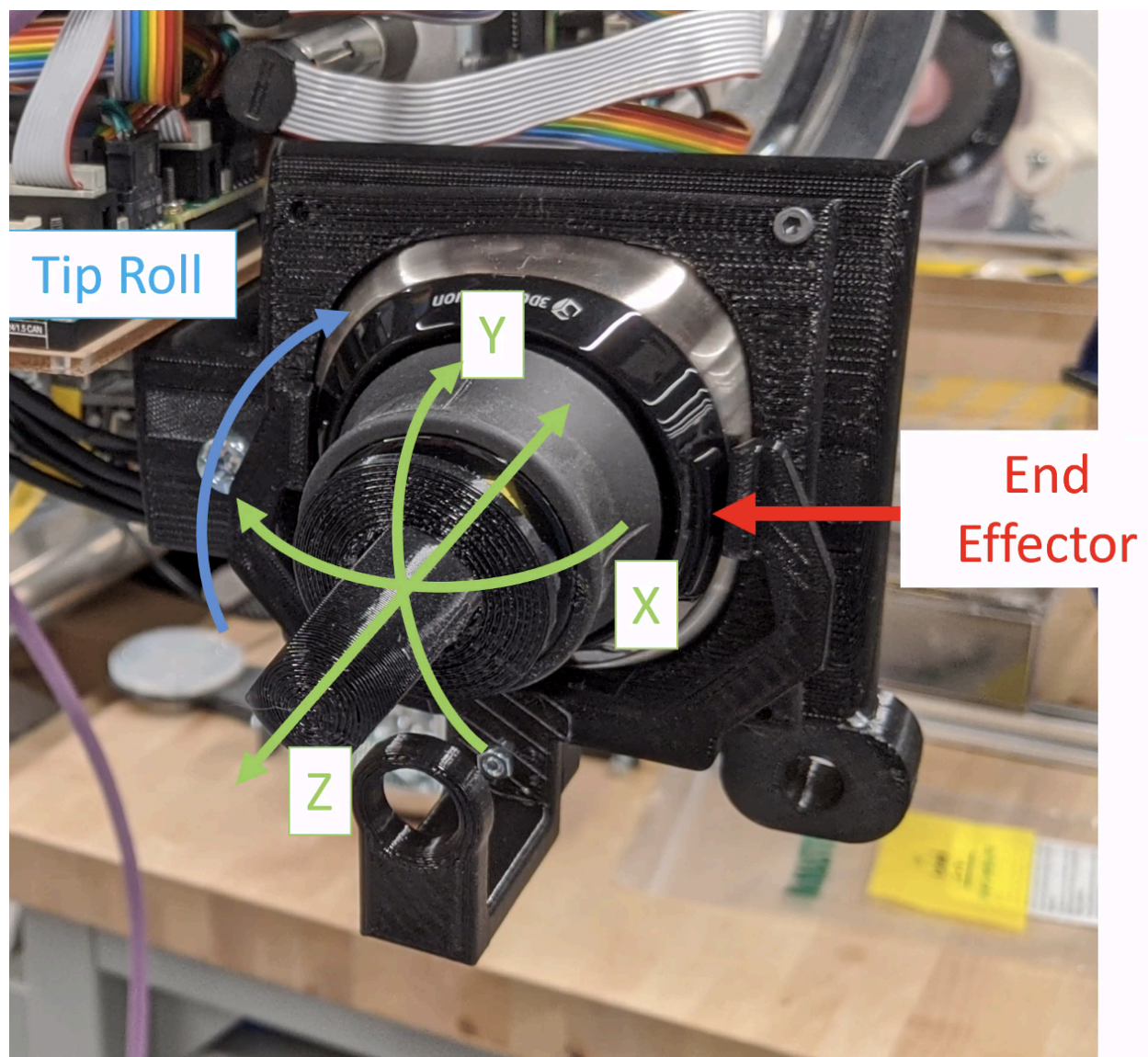


Overall System

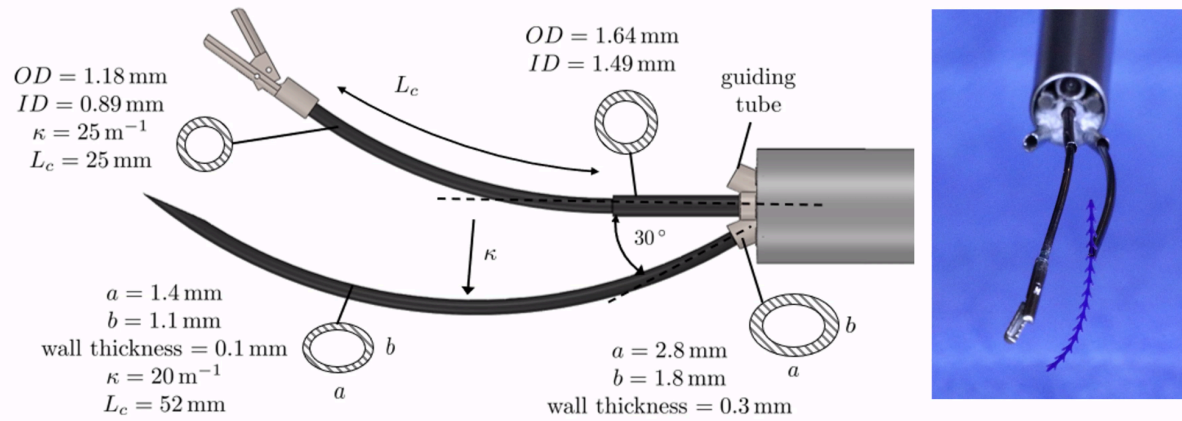


CTRs and
Endeffectors

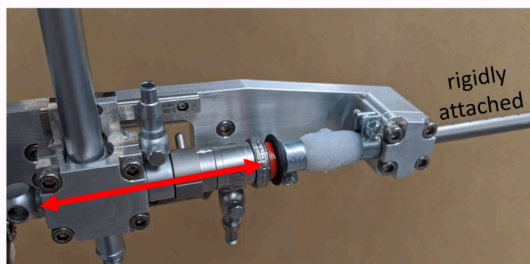
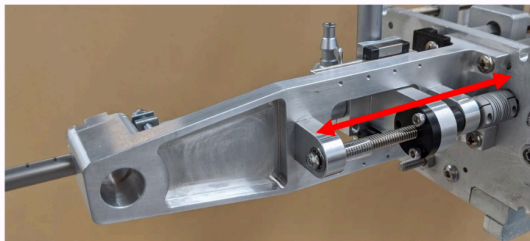
8/11/2021



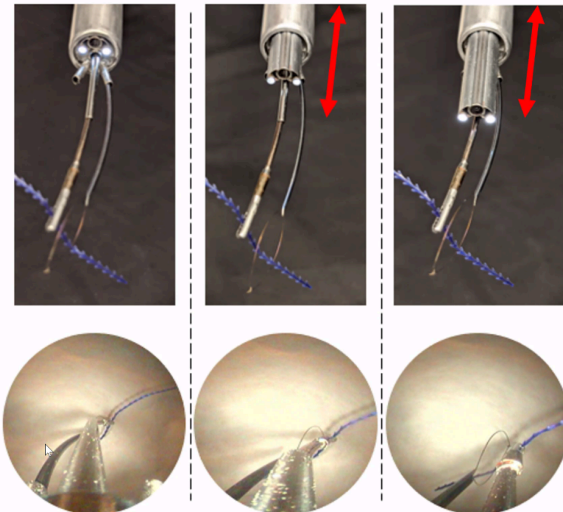
Manipulators First Generation



8/11/2021



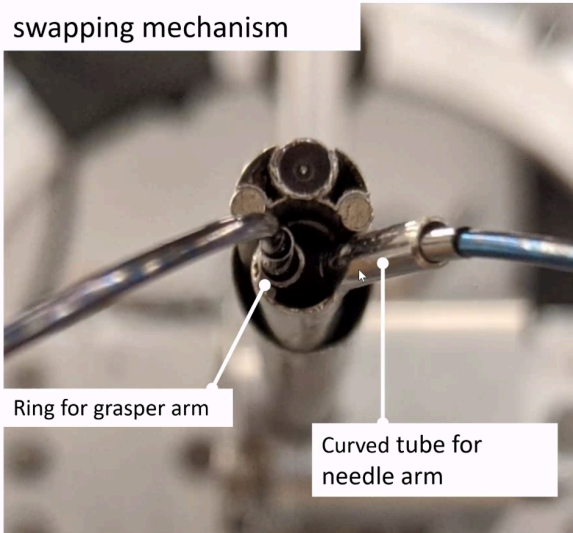
8/11/2021



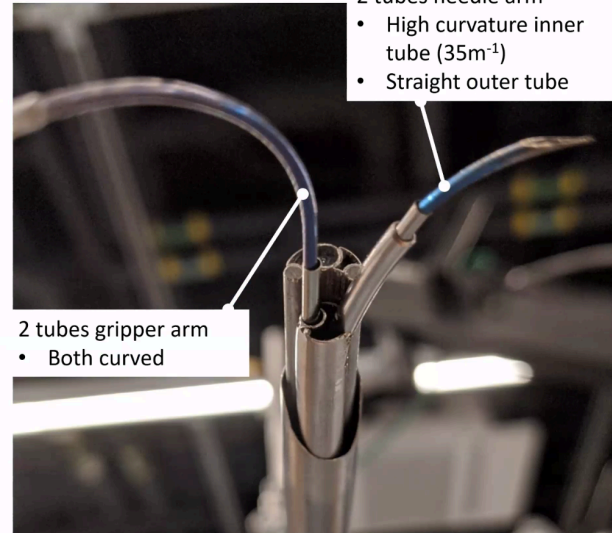
Manipulators Second Generation



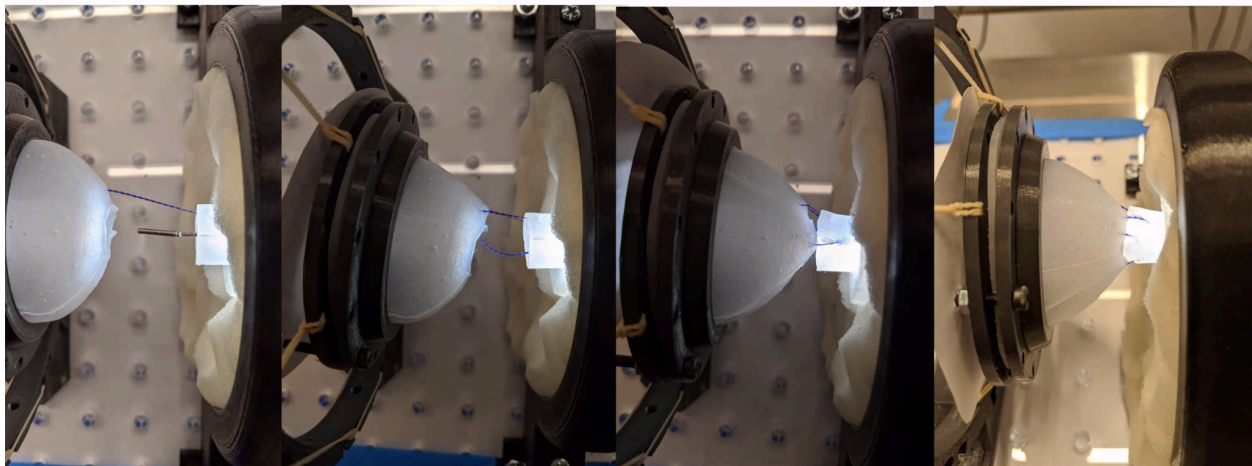
swapping mechanism



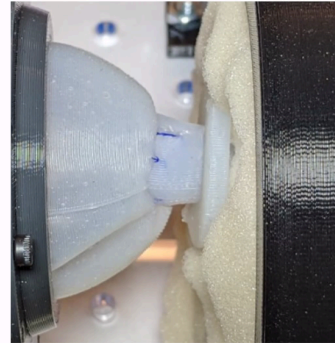
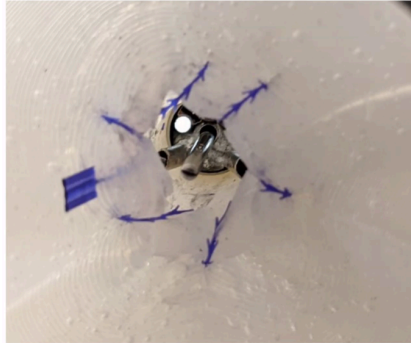
8/11/2021

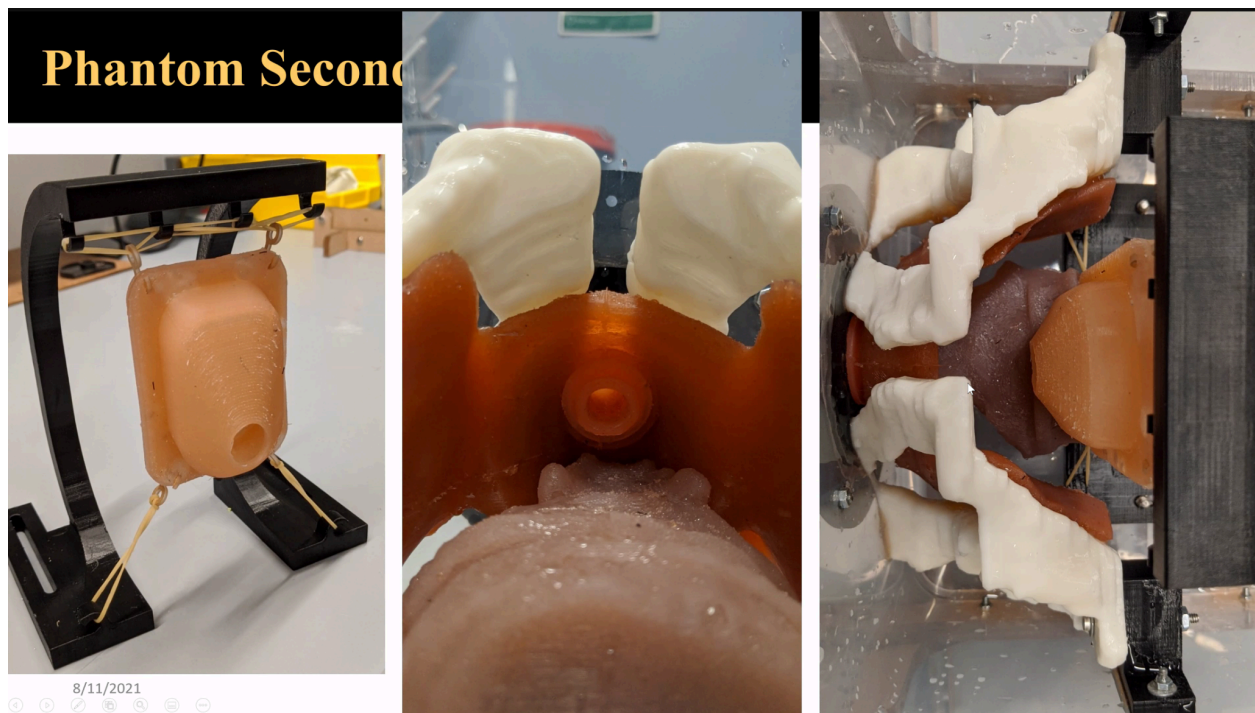
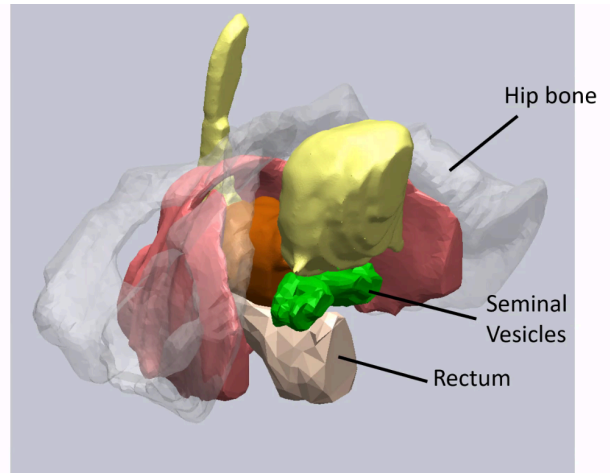
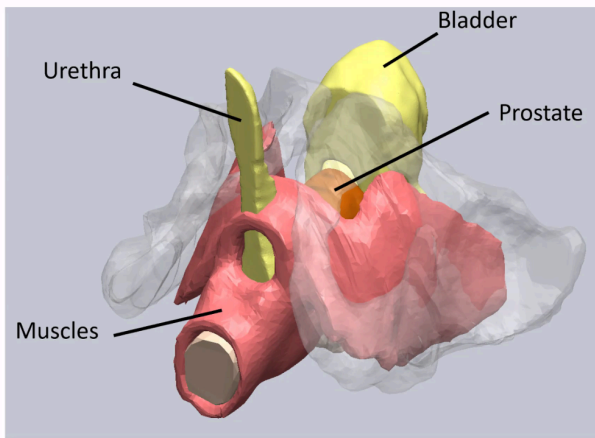


Results Hamlyn 2020

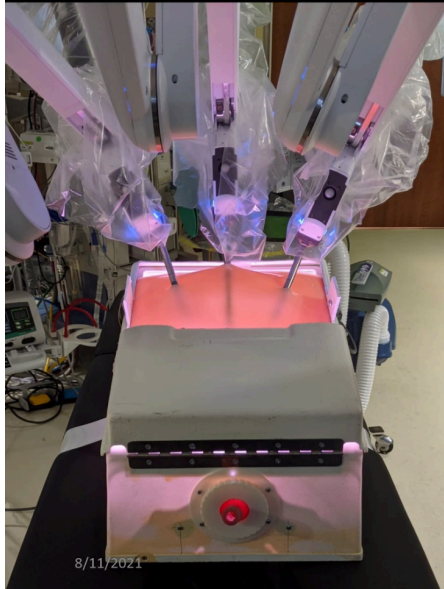


Results Hamlyn 2020





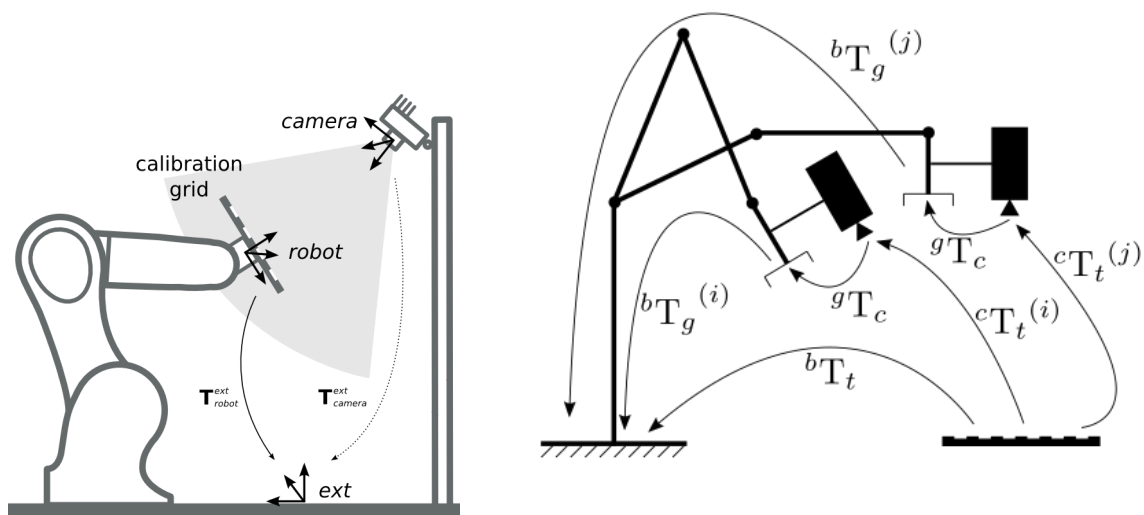
Phantom Second Generation - Evaluation



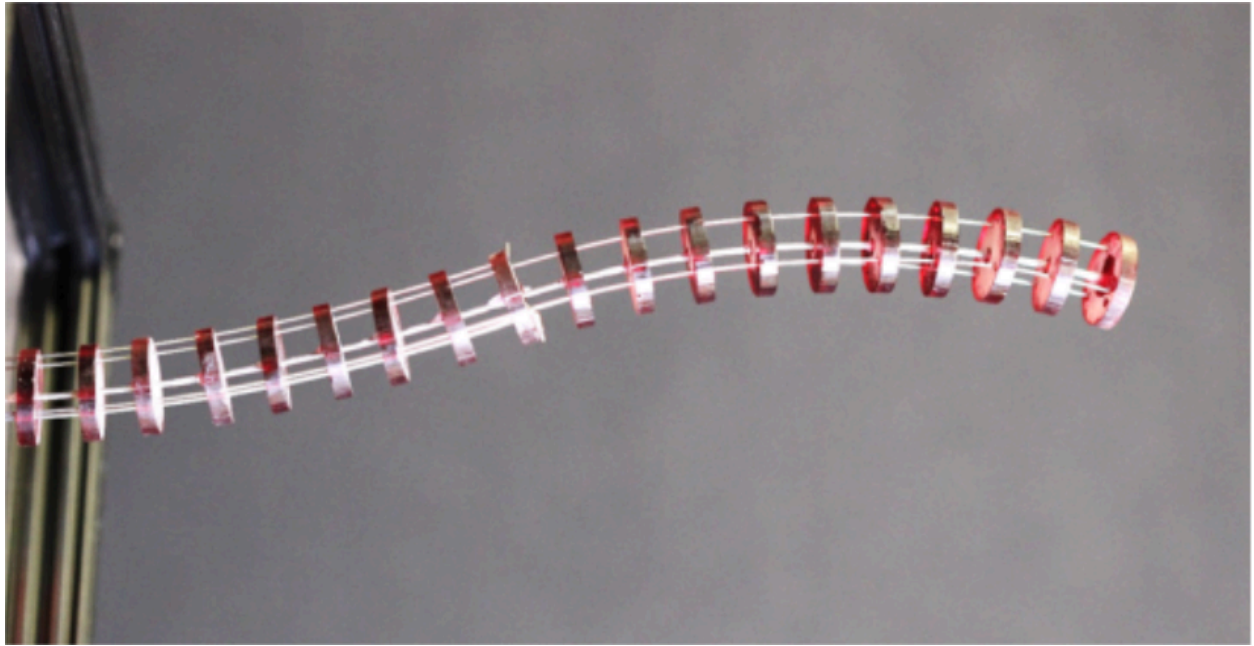
Phantom Squid



Extra Explorations (Built a Franka Robot w/ Reinhard! + made a timelaps)



This is the Hand-Eye calibration problem where the transformation between a camera (“eye”) mounted on a robot gripper (“hand”) has to be estimated!



En fin!