Robot Painter

Team Plum
Shubham Khandelwal, SungWook Lee, Chien-Ning Chen, Anthony Nguyen

Outline

- Introduction
- Robotic Arm
- Robotic Analysis
- System Setup
- Motion Generation
- Demo
- Alternatives
- Design Decisions

Motivation

- Painting more than just a combination of colors at a position.
- A digital representation is a description of (r,g,b) of every pixel. Recreating image from this or creating image based on this concept causes an artificial painting or ‘inorganic’ painting.
- It involves understanding that painting is more than a color value at position (x,y,z)
- Thus paint brushes can generate a larger variety of dynamic outputs by varying features like the angle, pressure on the tip and speed of the brush relative to the canvas.

Motivation

- Most drawing robots resemble plotters in a way as they are concerned about the (x,y,z) position of the drawing.
- Thus they tend to miss out the subtle variations that are possible by considering ways in which a stroke can be created.
- It will be beneficial to create a modular system by which a painting can be broken into small modules or strokes
- Thus a combination of such different available strokes can increase the organic factor of a painting

Problem Statement

- Create a robotic structure which can be used for creating more than a geometric/artificial painting by replicating the human hand motion.
- To be able to come up with a set of parameters using which different strokes in painting can be defined
- To be able to modularize approach to painting by defining a painting as a set of strokes and thus be able to create it by the robot as a combination of such strokes.

Project Changes

- Replicating hand generated motion on the robotic arm to avoid the pixel effect.
- Instead of parametrizing the strokes we plan to define strokes using suitable mathematical expressions
- Modularize a painting based on various available patterns instead of strokes
- Be able to create a painting based on these newly defined strokes, thus modularizing it.

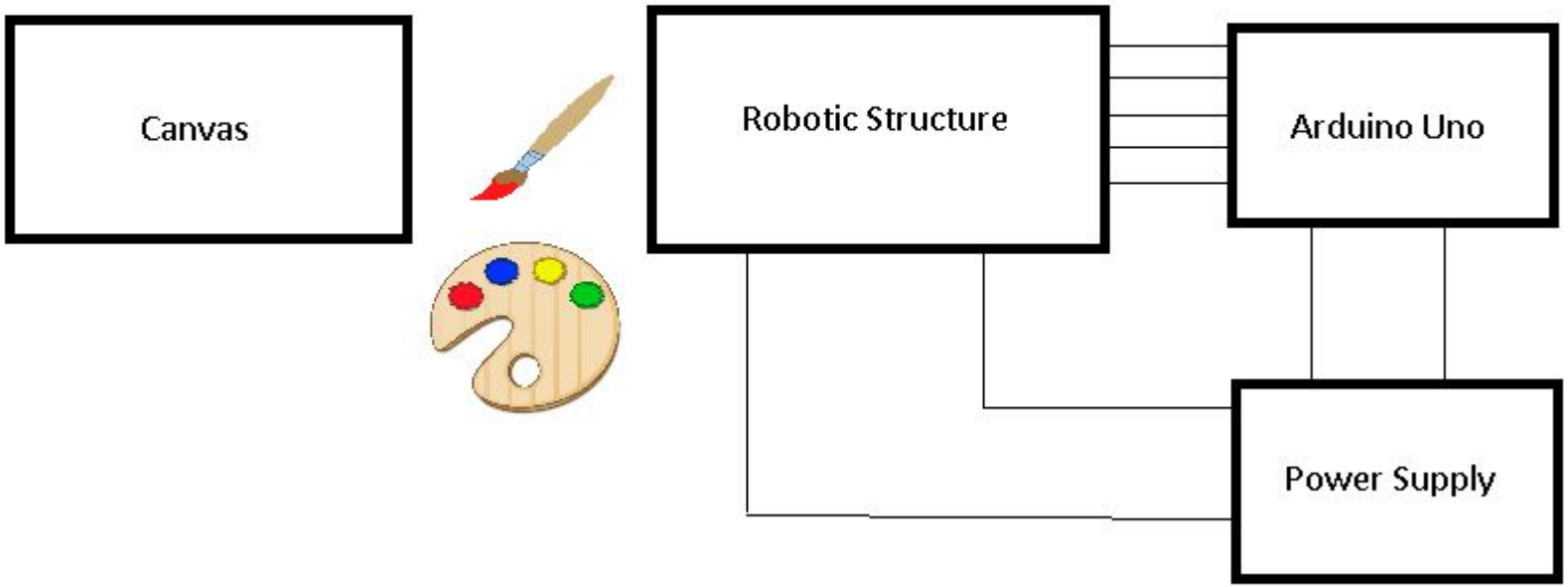
Scope

- The base frame is assumed to be fixed.
- The scope of our project is exclusively on usage of brush, possibly of more than one types.
- Our work mainly focuses on many different techniques of using brushes.
- If time is permitted, color mixing, color techniques (gradation) may be implemented.

System Design

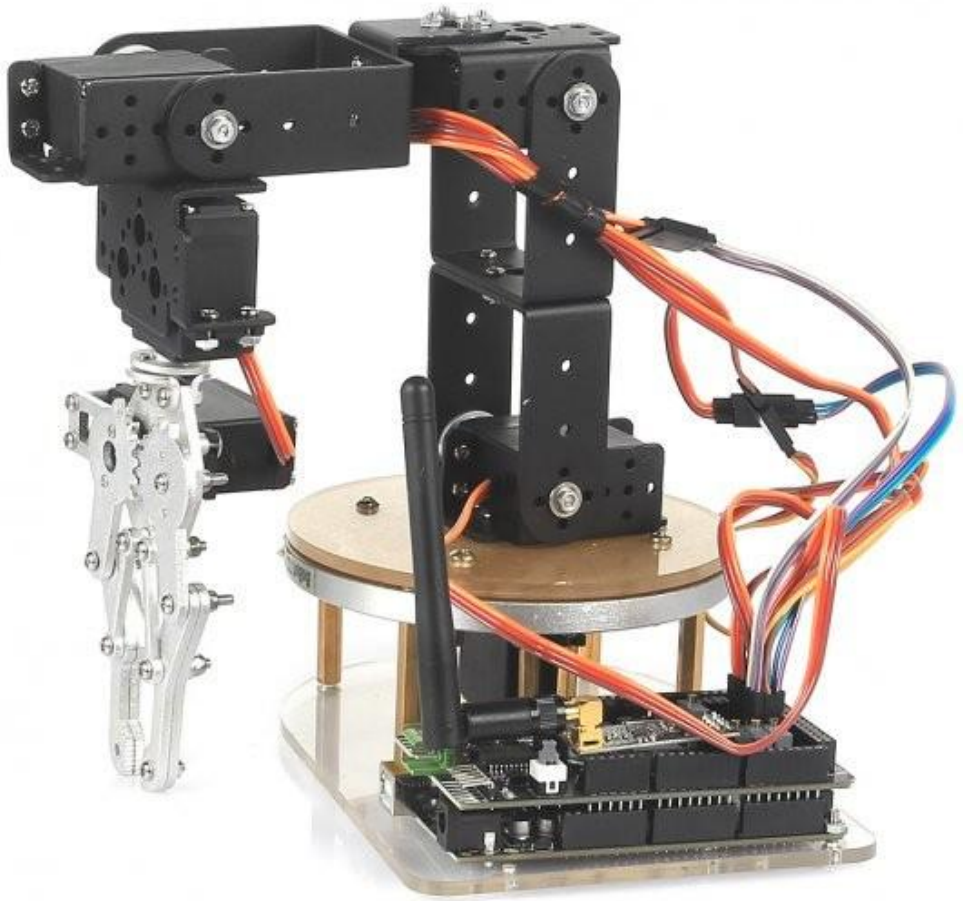
- The system basically has four components
  - The electrical system for power along with arduino for controlling the servo motors
  - The robotic arm structure
  - The paint brush with the color attached as an end effector to the robotic arm
  - The canvas

System Design

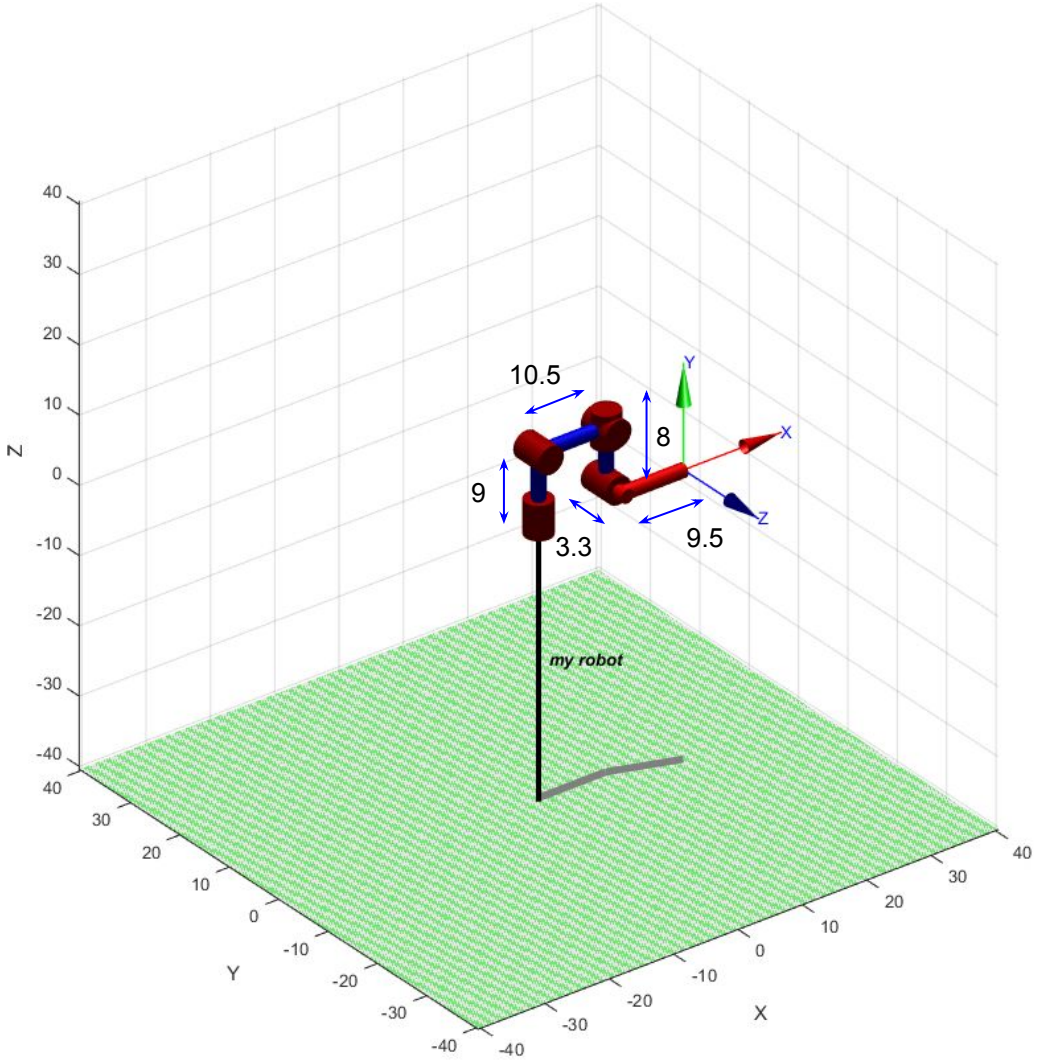


Robotic Arm Structure

- PVC material
- CNC processing
- 6 Servo Motors (6 - DoF)
- We use 5 motors
- Spherical Wrist
- Mounted on rotatable platform
- Gripper substituted with paint brush
- Closed loop with errors*



Robotic Arm Model



DH Parameters

- We have used the standard DH parameters

α	90°	0°	90°	-90°	0
d	9	0	0	8	3.3
a	0	10.5	0	0	9.5
θ	$\theta 1$	$\theta 2$	$\theta 3$	$\theta 4$	$\theta 5$

Calibration (Angle Adjustment)

- Done while assembling
- Arduino IDE, Matlab-Arduino Package, Yocto mraa library all has different angle standard
- Issues
  - Arctual angle VS pwm not linear
  - Combinations of angles to avoid (wall, floor, etc)
  - Angle resolution
  - Speed of angles

Robotics Analysis

- Forward Kinematics
- Inverse Kinematics
- Feedback control
  - Desired actual position VS what the arm thinks it is my desired actual position.

Forward Kinematics

- Given the transformation function (2.52, Bruino Chpt 2):
- $A_i^{i-1} =$

$$\begin{pmatrix} \cos(v_i) & -\sin(v_i)\cos(\alpha_i) & \sin(v_i)\sin(\alpha_i) & a_i\cos(v_i) \\ \sin(v_i) & \cos(v_i)\cos(\alpha_i) & -\cos(v_i)\sin(\alpha_i) & a_i\cos(v_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- $T_5^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 = \begin{pmatrix} Rotational\,(3X3) & Position(3X1) \\ 0(1X3) & 1 \end{pmatrix}$

Forward Kinematics

- Rotational Matrix

$$\begin{matrix} c_5(s_1s_4 + c_{23}c_1c_4) - s_{23}c_1s_5 & -s_5(s_1s_4 + c_{23}c_1c_4) - s_{23}c_1c_5 & c_4s_1 - c_{23}c_1s_4 \\ -c_5(c_1s_4 + c_{23}s_1c_4) - s_{23}s_1s_5 & -s_5(s_1s_4 + c_{23}c_1c_4) - s_{23}c_1c_5 & c_1c_4 - c_{23}s_1s_4 \\ c_{23}s_5 + s_{23}c_4c_5 & c_{23}c_5 - s_{23}c_4s_5 & -s_{23}s_4 \end{matrix}$$

- Point

$$\begin{matrix} \frac{33c_4s_4}{10} + \frac{19}{2}c_5(s_1s_4 + c_{23}c_1c_4) + \frac{c_1(16s_4 + 21c_2)}{2} - \frac{33c_{23}c_1s_4}{10} - \frac{19s_{23}c_1s_5}{2} \\ \frac{s_1(16s_{23} + 21c_2)}{2} + \frac{19}{2}c_5(c_1s_4 + c_{23}s_1c_4) + \frac{33c_1c_4}{10} - \frac{33c_{23}s_1s_4}{10} - \frac{19s_{23}s_1s_5}{2} \\ \frac{21s_2}{2} - 8c_{23} - \frac{33s_{23}s_4}{10} + \frac{19s_{23}c_4c_5}{2} + 9 \end{matrix}$$

Inverse Kinematics

- Closed form

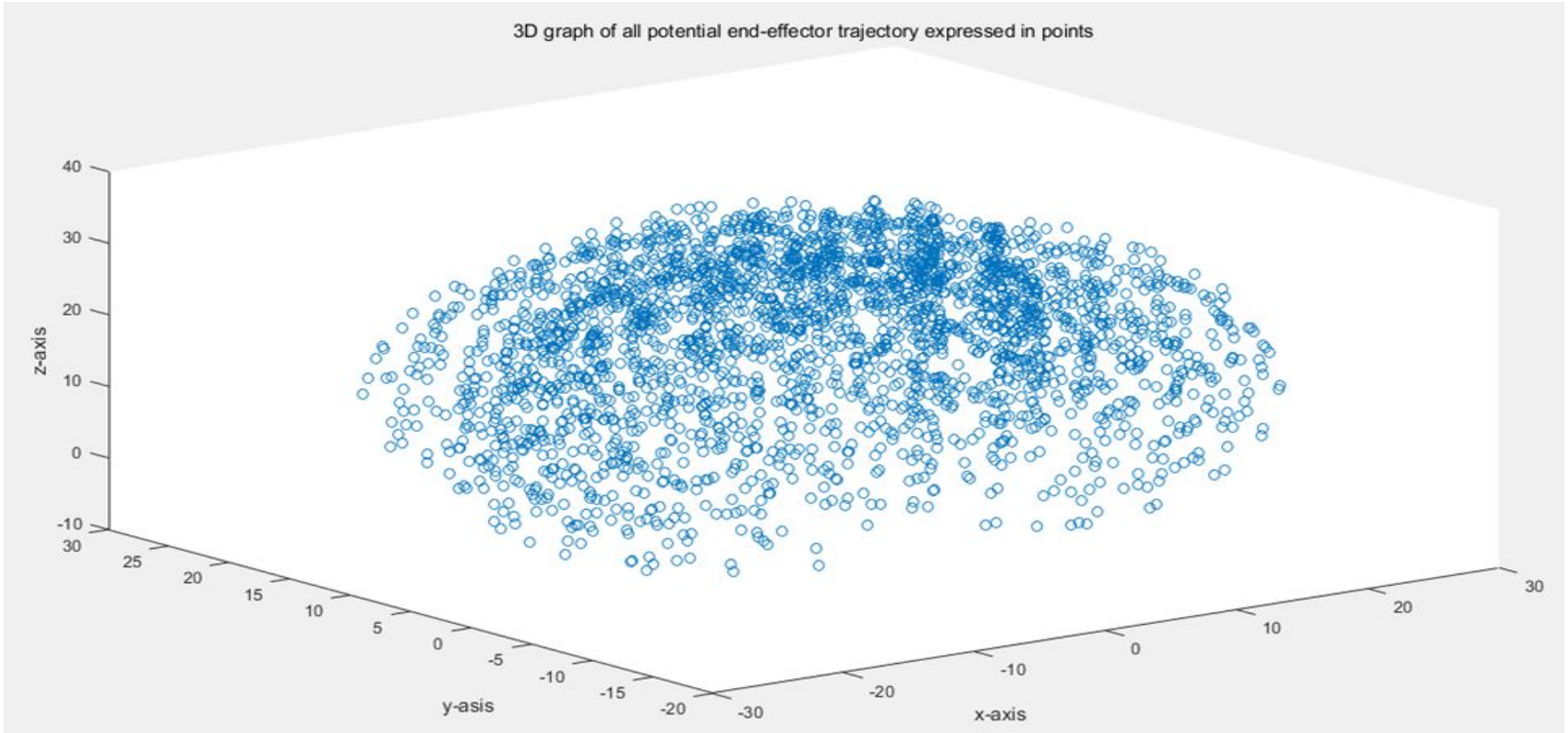
- Using Optimization Method
  - Optimizing objective function:
    minimize (inv(T)*robot.fkine(q) - eye(4)) * omega)
  - Use “fmincon” function in Matlab

Inverse Kinematics Algorithm

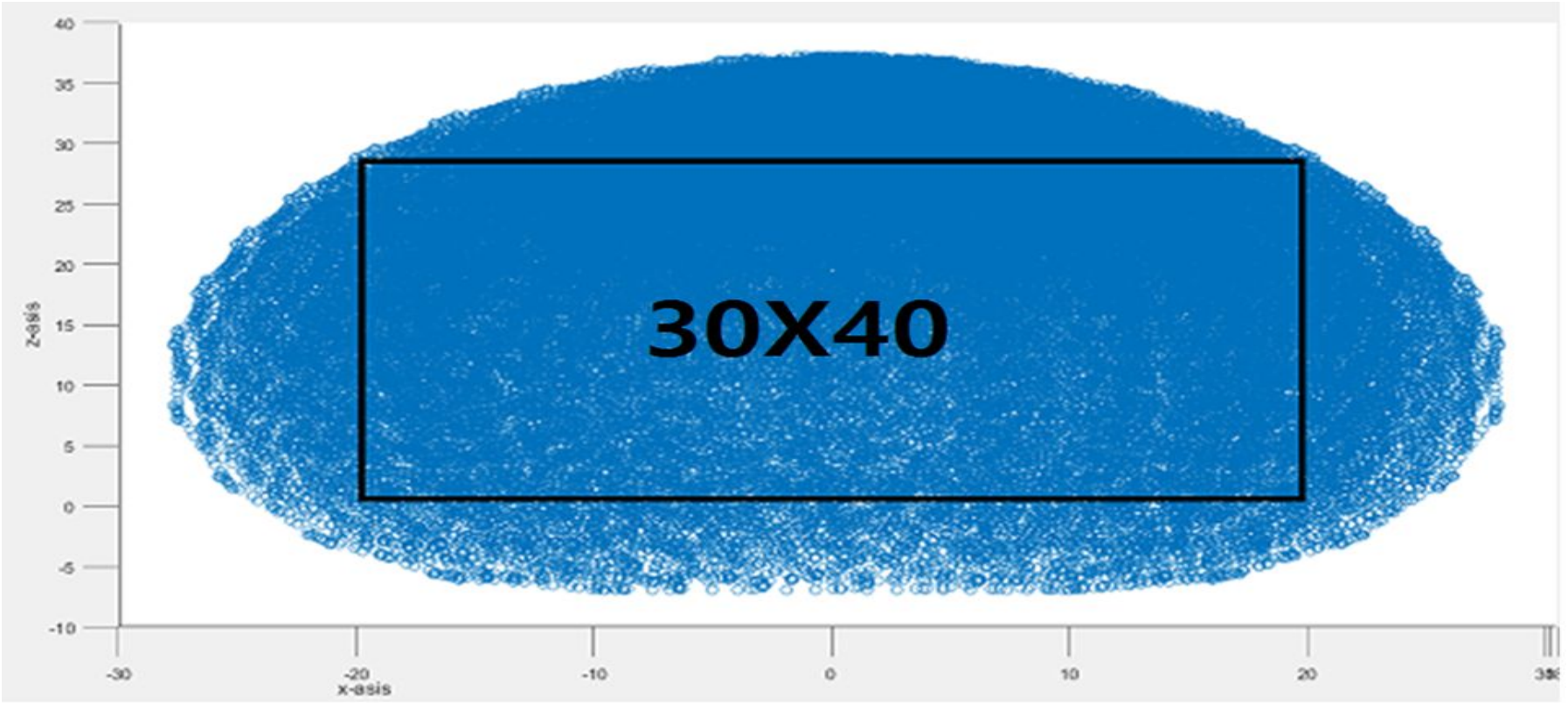
- Using Jacobian matrix
  - Use analytical Jacobian (Pseudo inverse, transpose) and iteratively reduce the x,y,z, and Euler angles

1. Enter initial point, initial angle
2. Find target point, initial error
3. If the point is within the reachable space, start the loop.
4. Find analytical Jacobian at current point
5. Let $\Delta Q = K^* (J_{numeric})^T * error$
6. Let $Q = Q + \Delta Q$
7. Update new position with new Q
8. Find new error: new position – target position
9. Repeat 4~8 until desired norm(error) is achieved

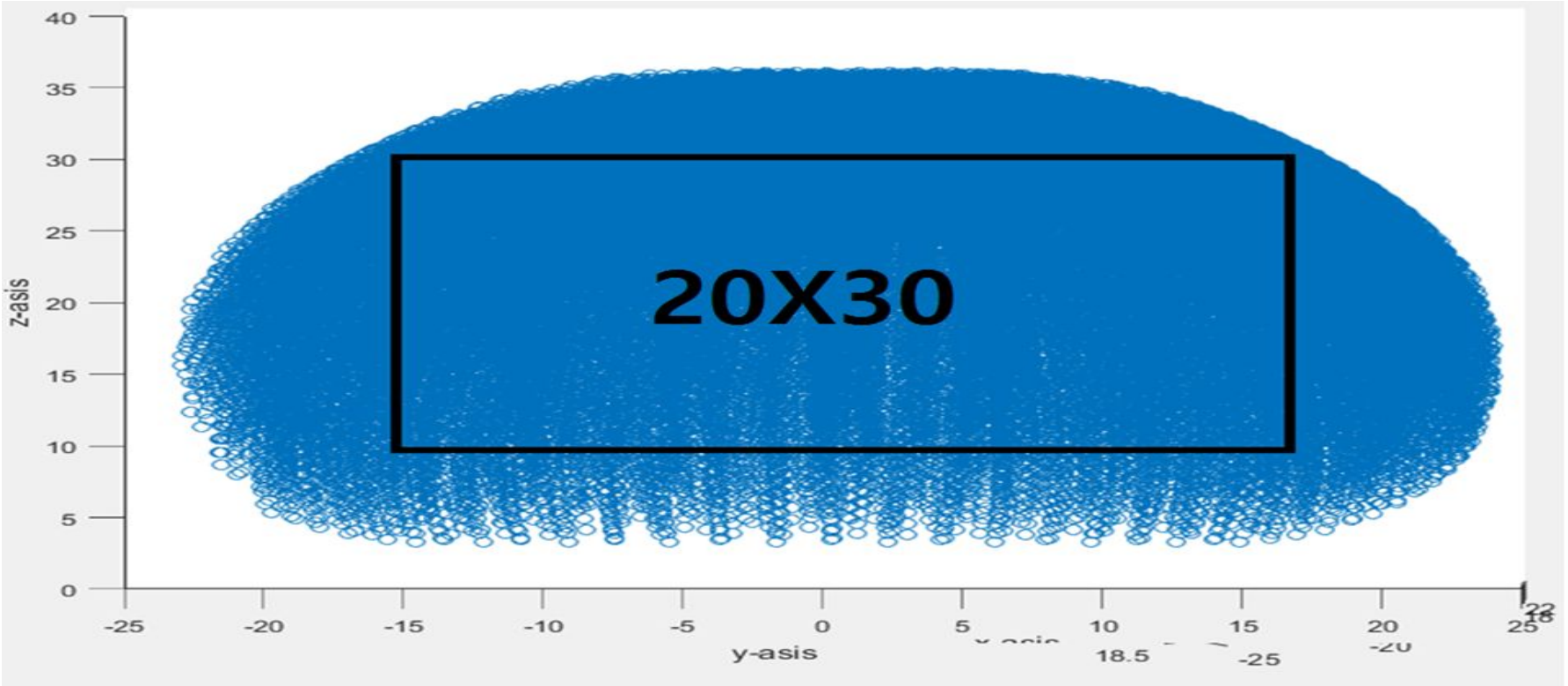
Reachable Operational Space in 3D



Reachable at plane y = 19.5cm



Flexible space at plane y = 19.5cm

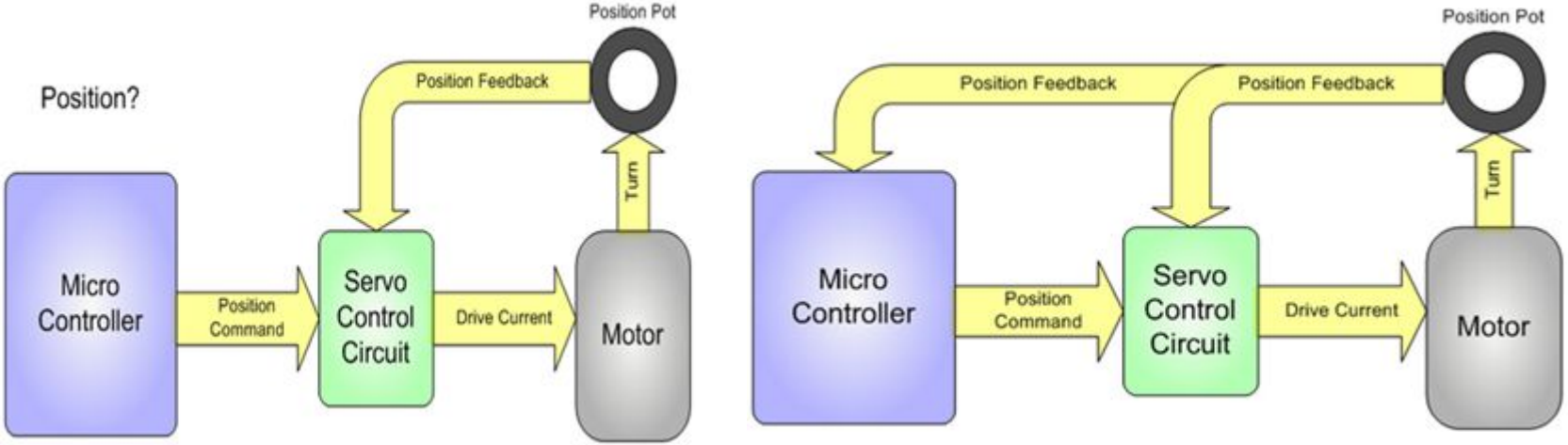


Reasons for inaccuracy

- Accuracy/errors of inverse kinematics
  - Efficient and accurate IK is still an active area of research
- Firmness of motor shaft
- Weight (of each joints)
- PWM sensitivity (PWM resolution)
  - Cannot account for precise angle assignment
- Misuse or lack of feedback

Feedback

- Why do we need feedback?

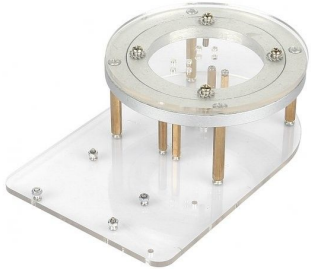


- The angle that servo think it is at may not be the same as the actual angle we see.
  - electrical interference, physical interference, weight of joints, etc

(https://learn.adafruit.com/analog-feedback-servos/about-servos-and-feedback)

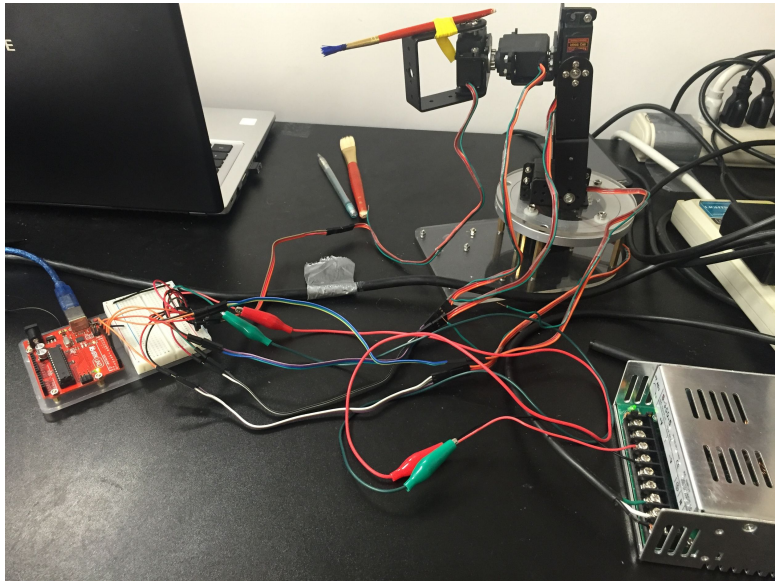
Hardware Design

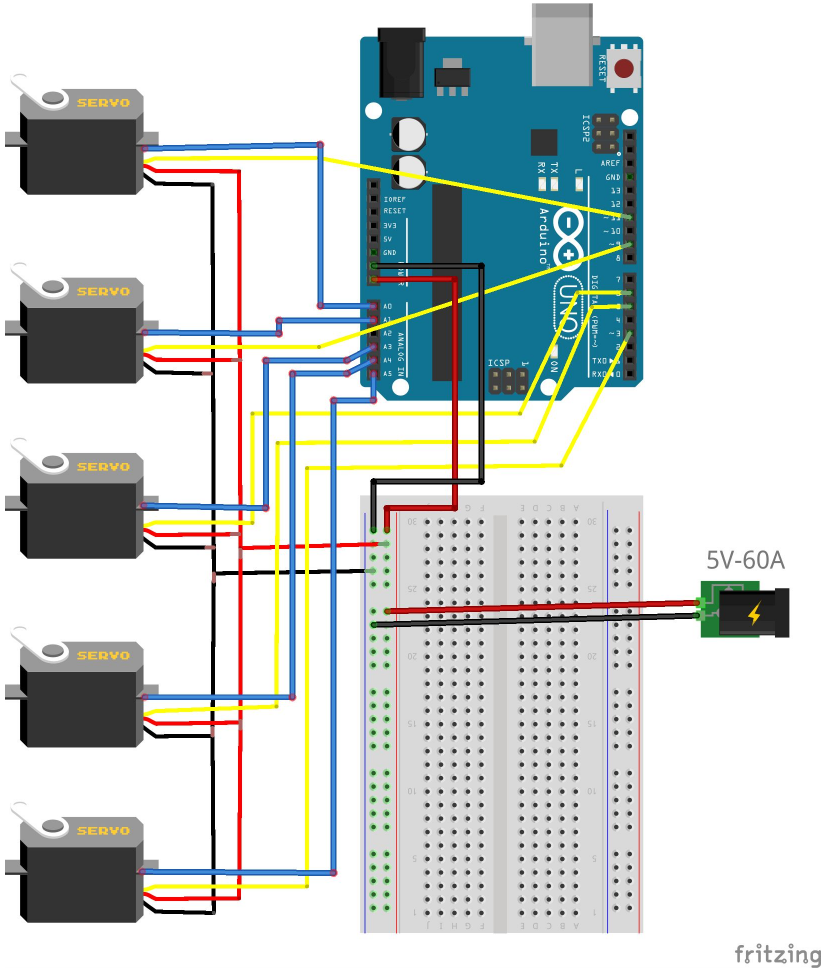
- Arduino UNO
  - Enable up to 6 concurrent PWM outputs.
  - Fully compatible with Matlab.
- Robotic arm with rotatable platform
- PWM-controlled Servo
  - Each one contributes 1 DOF to the operational space.
  - Hack each servo to obtain its internal potentiometer reading as a feedback reference.
- External Power Supply (5V-60A)
  - Prevent servo from draining excessive current from Arduino.
  - (Decoupling capacitor)



Circuit Diagram

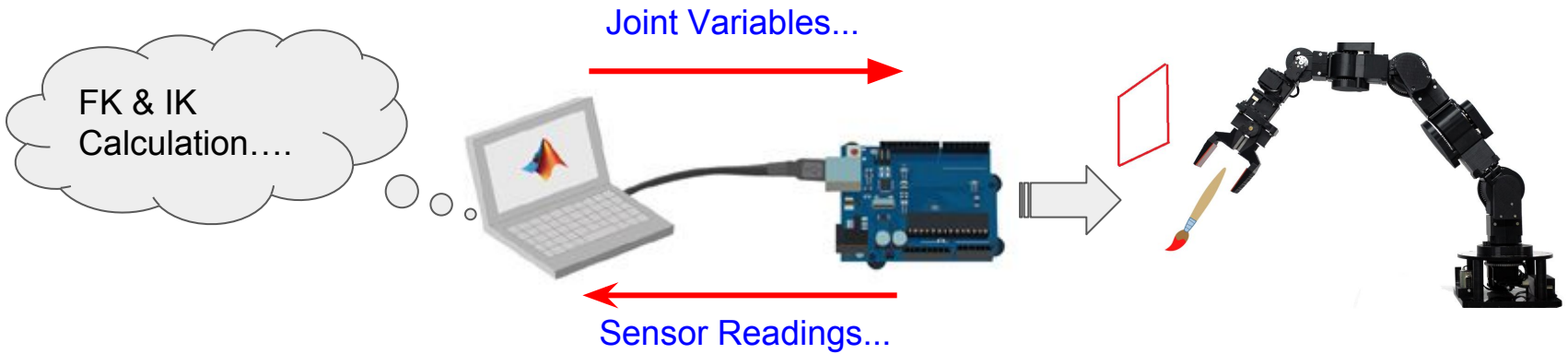
- Arduino must be grounded with power supply to guarantee the same voltage level for signals.





Arduino - Matlab Integration

- Matlab Arduino Package
  - Provide interactive communication over the USB without compiling.
  - Direct control over all peripherals. (Digital, Analog, PWM, I2C...etc.)
  - Process retrieved data directly with Matlab function.
- The computation overhead can be off-loaded to the Matlab on a host laptop.
  - e.g. Forward kinematics and Inverse kinematics.



Motion Generation in Matlab

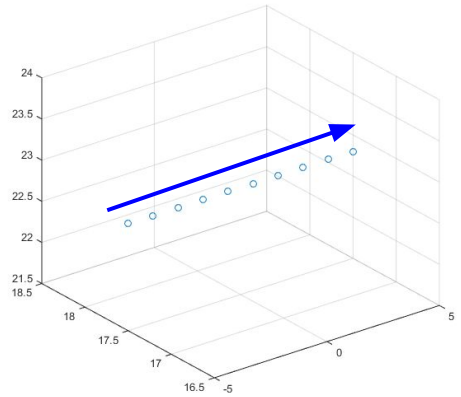
- Programmatically generate end-effector coordinates in Matlab.
  - Horizontal Straight Line
    - Linear space
    - Logarithmic space
  - Vertical Straight Line
  - S Stroke
  - Semi-Circle
  - …
- Main tuning features for a stroke motion:
  - **Trajectory**
  - **Speed**
  - **Granularity**
  - **Angle**

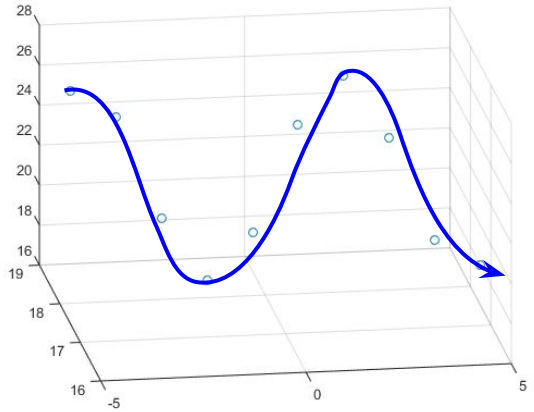
Motion Generation in Matlab

- Use various mathematical functions to generate different trajectories.
  - Horizontal Straight Line: **X = linspace(-5, 5, 10);**
  - S Stroke:

    X = linspace(-5, 5, 10);

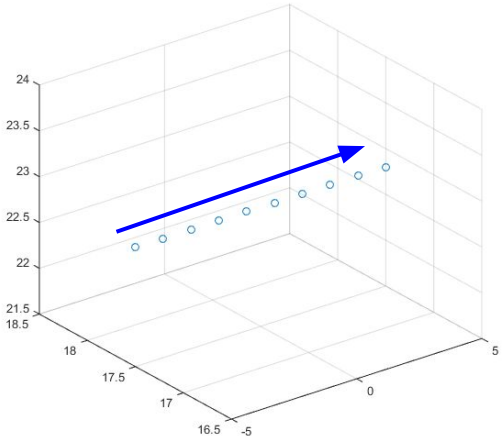
    Z = 22.8 + 5*sin(X);
- Equivalently mimic brush “**Trajectory**” variant!

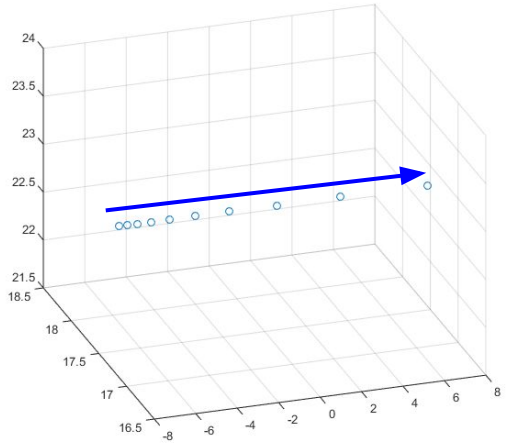




Motion Generation in Matlab

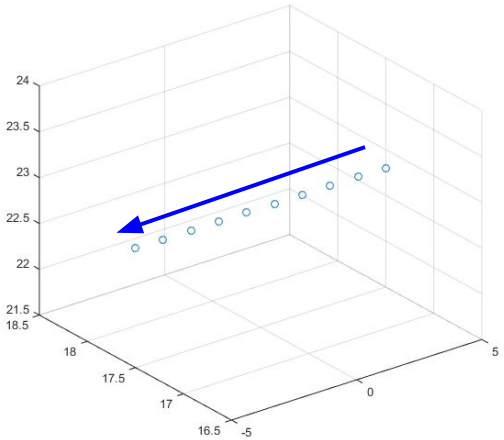
- Use unequal point spacing to produce brush speed difference.
  - Horizontal Straightline
    - Linear: **X = linspace(-5, 5, 10);**
    - Logarithmic: **X = logspace(1.2, 0 , 10);**
- Equivalently mimic brush “**Speed**” variant!

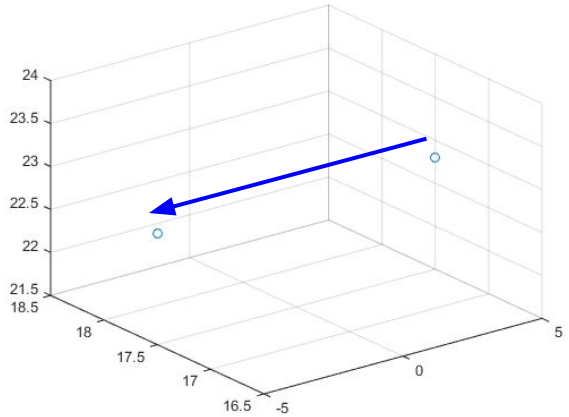




Motion Generation in Matlab

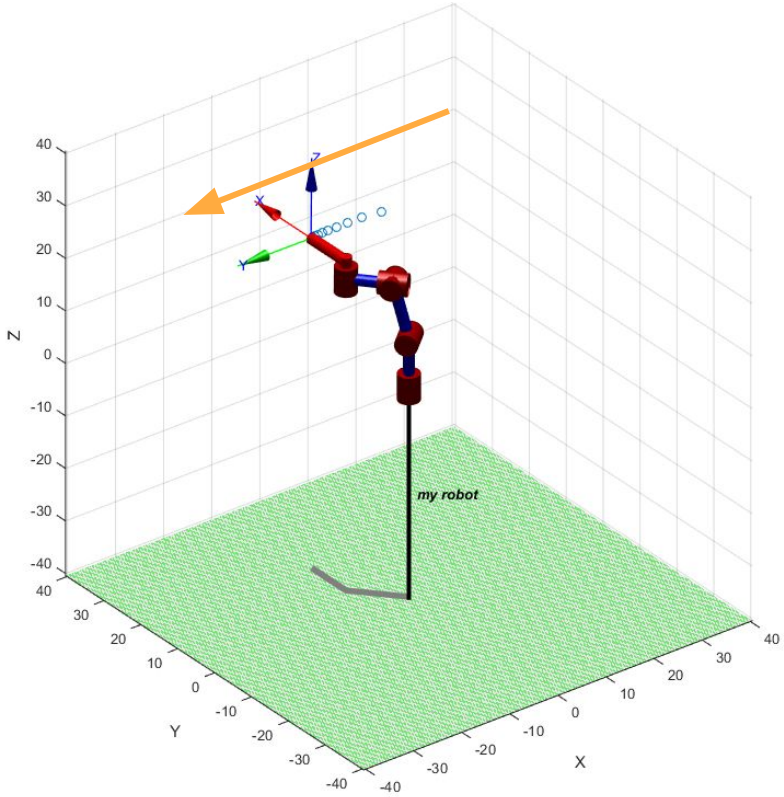
- Use fewer points in a stroke to decrease the trajectory granularity!
  - Horizontal Straightline
    - 10 Points: **X = linspace(-5, 5, 10);**
    - 2 Points: **X = linspace(-5, 5 , 2);**
- Equivalently mimic brush “**Granularity**” variant!





Motion Simulation in Matlab

- Robotics Toolbox
  - Available in Matlab.
  - http://petercorke.com/Robotics_Toolbox.html
- Animate brush trajectory with known joint variables.
- Verify planned motion before actual deployment to the servo.

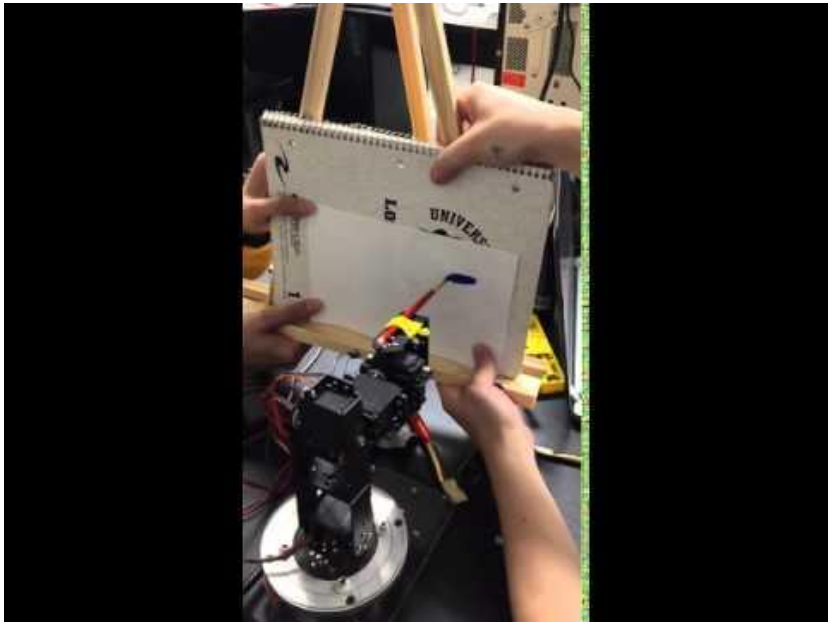


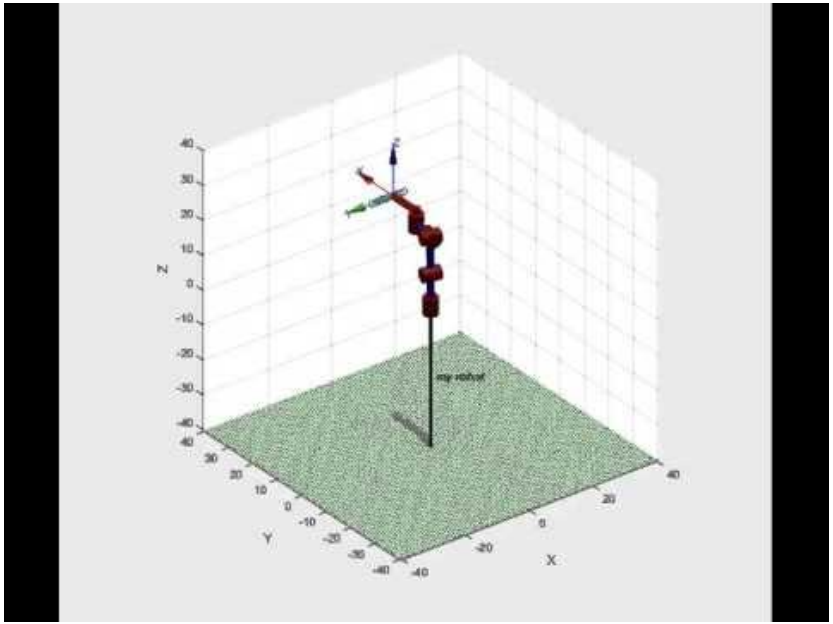
System Evaluation

- We generate the required motion sketch based on various inverse kinematics solutions available at our disposal
- The motion is then simulated using the toolbox to evaluate it for errors and check expected behaviour
- We also create a plot of the trajectory of the brush to see the accuracy of the inverse kinematic solutions
- This also helps in understanding the overall system accuracy
- We then execute this motion on the robotic arm and check the angles of the motors continuously using feedback
- The motion is then used for actually painting on the canvas

Demo 1

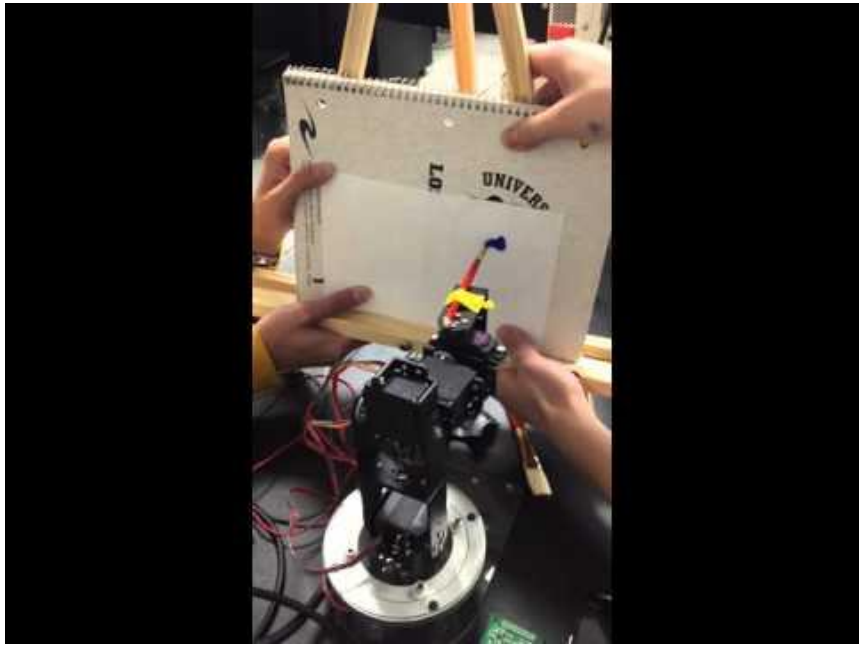
- Horizontal Straight Line: Linear Space

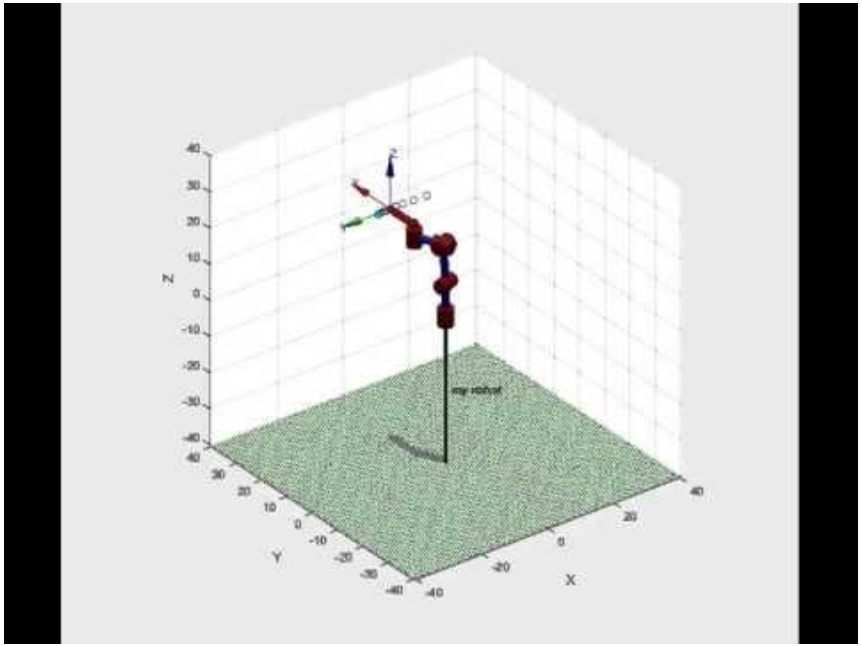




Demo 2

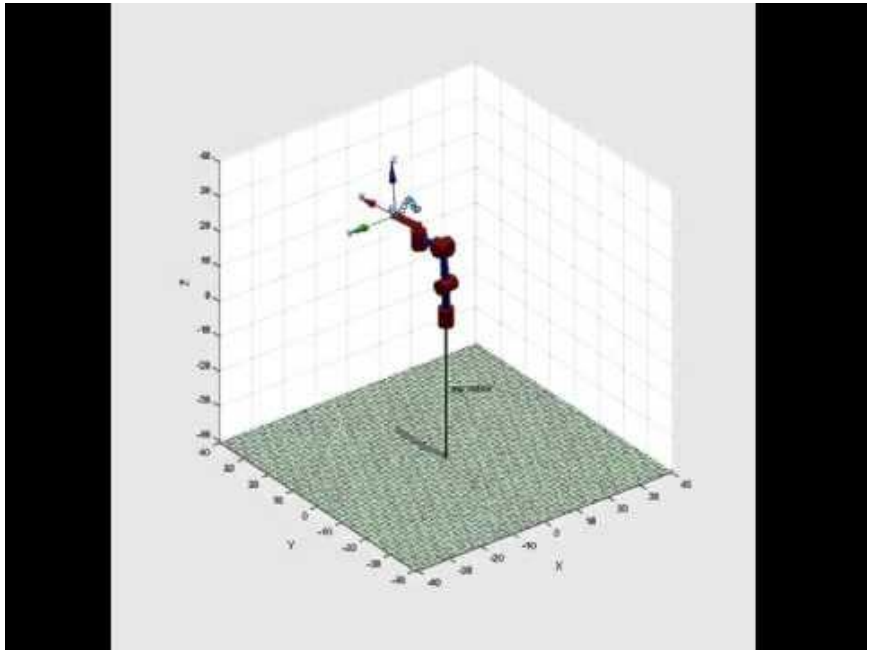
- Horizontal Straight Line: Log Space





Demo 3

- S Stroke:

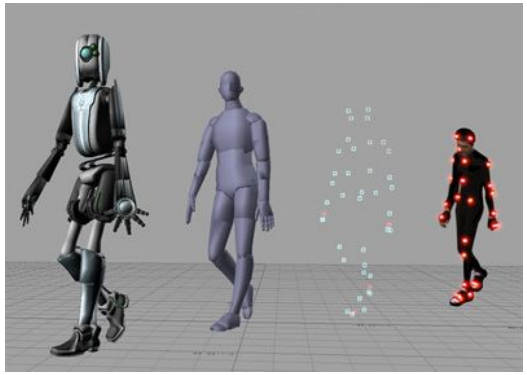


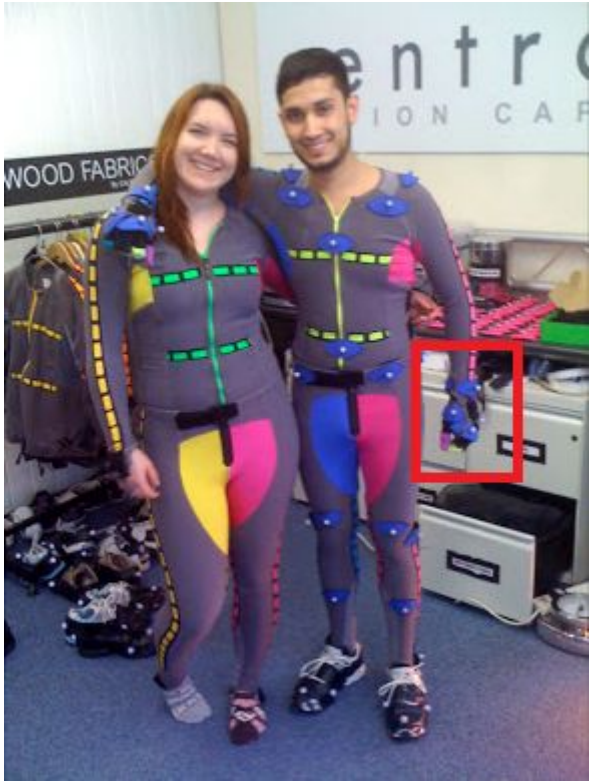
Motion Generation Alternatives (1)

- In addition to the MATLAB based stroke generation, I was inspired by the calligraphy group's mission - preserving national level art talent
- To do this, would need to somehow record the painting behavior of an artist, and then translate the behavior so our arm can reproduce it
- The most basic idea would be to record a video of an artist's arm motions as they painted, do image recognition on the arm motions and turn that into our robot's arm motion
    - This is a potential future direction but we did not think it made sense to do it with only a week left

Motion Generation Alternatives (2)

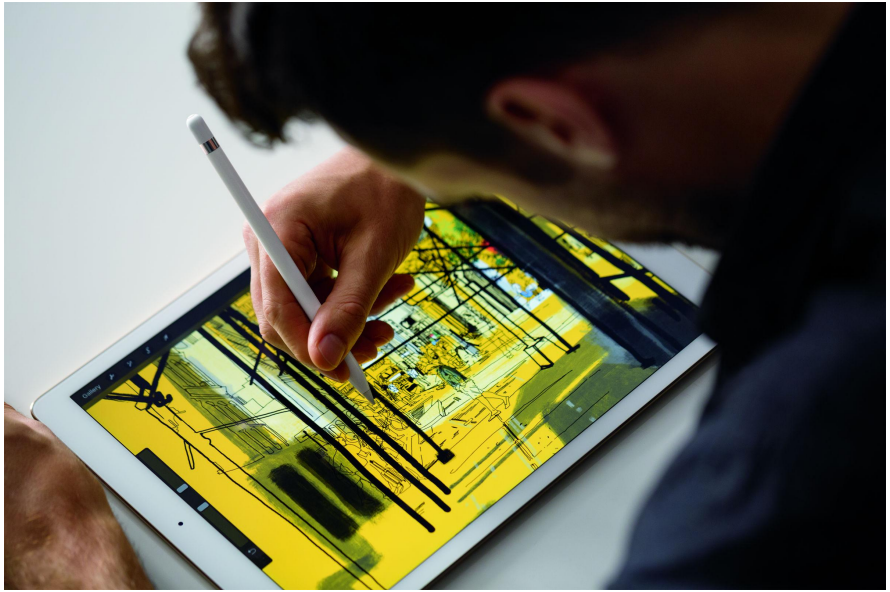
- True Hollywood Style Motion Capture!
- Special body equipment with recording system can capture motion in great detail
- Also not feasible
- So what IS feasible?





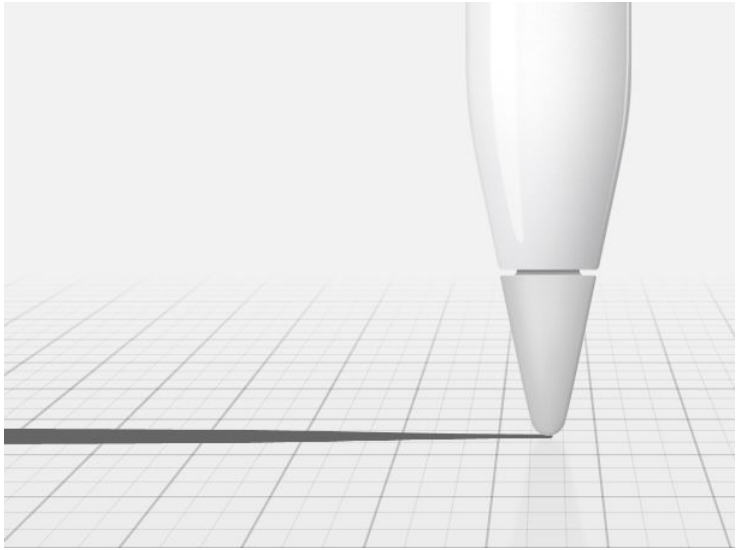
Motion Generation Alternatives (3)

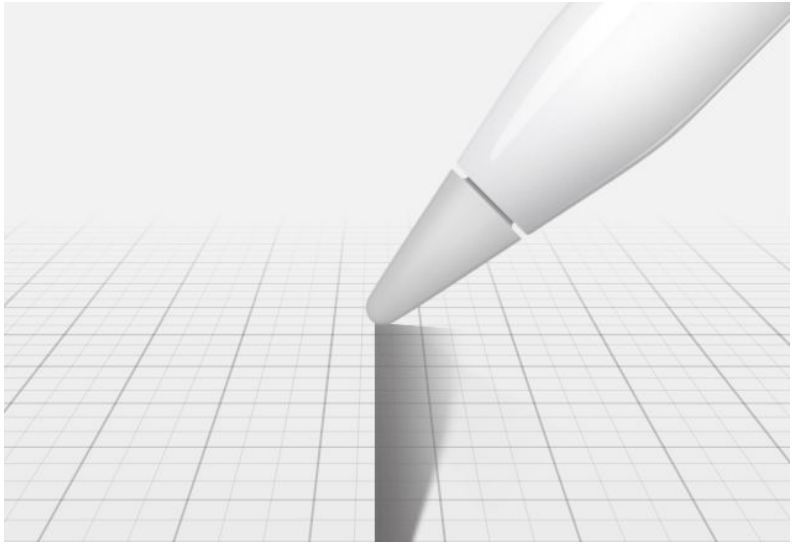
- Active Digitizer Technology
- Used by Wacom, N-Trig, and now Apple
- Drawing Tablets are used by consumers and professionals to create art, so we can use it for our robot as well
- Apple Pencil pictured here:



Motion Generation by User Interaction (1)

- What makes this an improvement over the use of plotters?
- Pressure sensitivity and tilt sensing

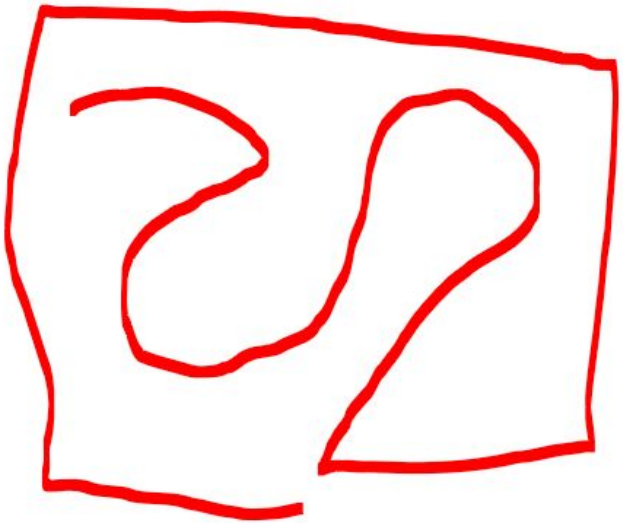




Motion Generation by User Interaction (2)

- My tablet doesn’t use the Apple Pen, so no tilt sensing, pressure only

Digital Sketch



Raw Data

1	71.99408	93.99371
2	71.99408	93.82023
3	71.99408	93.42244
4	72.16114	92.98013
5	72.55681	92.59074
6	73.16657	92.29401
7	73.77972	91.92058
8	74.46381	91.57228
9	75.23413	91.1287
10	75.91661	90.53941
11	76.60621	89.97743
12	77.35561	89.52466
13	78.17457	89.20345
14	78.88908	88.83559
15	79.60339	88.50693
16	80.36322	88.25375
17	81.19012	88.08134
18	81.90997	87.80516
19	82.62099	87.52403
20	83.38019	87.28822
21	84.20148	87.11517
22	85.08768	87.0029
23	85.85567	86.77306
24	86.60107	86.52351
25	87.37549	86.30653
26	88.20625	86.14272
27	89.09102	86.03334
28	89.85238	85.80271

763	323.1083	84.88919
764	324.0414	84.59261
765	325.0107	84.33982
766	326.0005	84.15192
767	326.9992	84.0284
768	328.0061	83.95806
769	329.013	83.92628
770	330.0222	83.91917
771	330.861	83.75856
772	331.639	83.54849
773	332.4279	83.34941
774	333.2552	83.18967
775	334.1344	83.07674
776	335.0573	83.00643
777	336.0186	82.96938
778	337.0027	82.95525
779	337.8309	83.12204
780	338.6139	83.35094
781	339.4081	83.57199
782	340.2401	83.75153
783	340.9495	84.05328
784	341.6505	84.36482
785	342.3951	84.6283
786	343.035	84.99002
787	343.6873	85.33936
788	344.4019	85.62313

1637	213.5369	33.37348
1638	212.7076	33.20268
1639	211.8301	33.08149
1640	210.9042	33.0057
1641	209.9444	32.9655
1642	209.1308	32.77644
1643	208.3616	32.5451
1644	207.5719	32.33194
1645	206.7391	32.16412
1646	205.8528	32.04748
1647	204.9235	31.97627
1648	203.9571	31.93988
1649	203.1365	31.76003
1650	202.3612	31.53903
1651	201.5668	31.33502
1652	200.731	31.17419
1653	199.843	31.06229
1654	198.9128	30.99388
1655	197.9526	30.95884
1656	196.9672	30.94645
1657	195.9696	30.94748
1658	194.9621	30.95505
1659	193.9537	30.96461
1660	192.943	30.97353
1661	191.9358	30.98061
1662	190.9281	30.98558
1663	189.9244	30.98864

2263	211.0494	369.989
2264	212.0103	369.9472
2265	212.9946	369.931
2266	213.9967	369.9303
2267	215.0043	369.9376
2268	215.8435	369.7805
2269	216.6293	369.5679
2270	217.4218	369.3637
2271	218.2564	369.1985
2272	218.9703	368.9073
2273	219.6694	368.6027
2274	220.4094	368.3431
2275	221.2174	368.1501
2276	222.0892	368.0233
2277	222.8447	367.7841
2278	223.5871	367.5293
2279	224.3637	367.31
2280	225.199	367.1458
2281	226.0886	367.037
2282	227.0276	366.9745
2283	227.8317	366.9457

Demo 4

Data To MATLAB

- After creating a single stroke, the stroke is exported into CSV format
- Matlab can directly import CSV format, however the data can’t be used directly by the arm
- Centering, Scaling, Sampling, and Calibration is required to produce motion

Paint Design Decisions

- Literature survey did not reveal papers characterizing stokes
- Instead, a visit to the Blick Art Supply store helped enlighten us to some of our design decisions around our paints

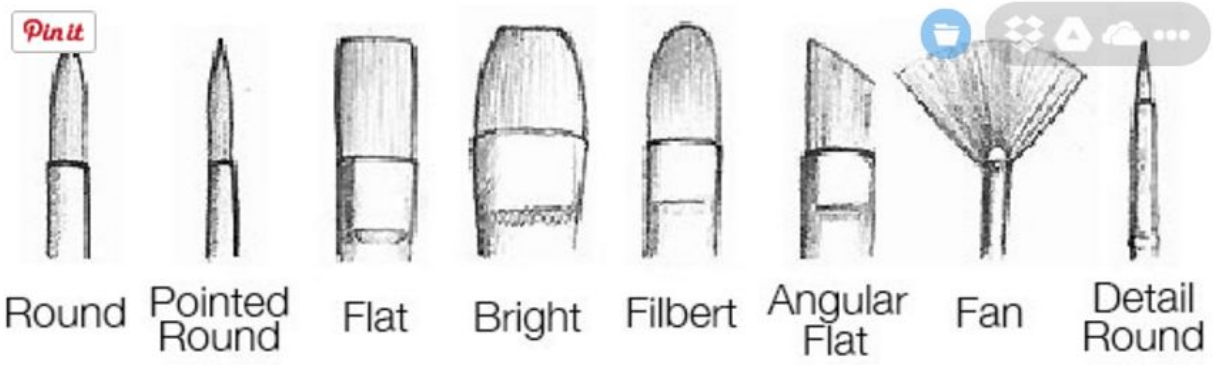
Choice of Paint: Acrylic

- Acrylic
  - water based/soluble; easy to clean and work with
  - dries quickly
  - sticks to most surfaces
  - possible to paint layers atop each other immediately
  - less runny than watercolors
- Oil Paints
  - resists water
  - require oils to thin or thicken the paint body
  - requires more time to set up
  - more expensive
  - may take days to dry

In the end, we chose acrylic because it made more sense to focus our project on the robot motion, and not emphasize other aspects of art such as mixing the paints

8 Main Types of Paintbrushes

- Suggested to use Flat and Round
  - Round brush will produce the same stroke regardless of the angle the brush is held at - most basic brush of any artist, told we could not skip it
  - Flat brush is rectangular, can create two fundamental strokes, one for length, one for width, and by rotating other strokes can be obtained

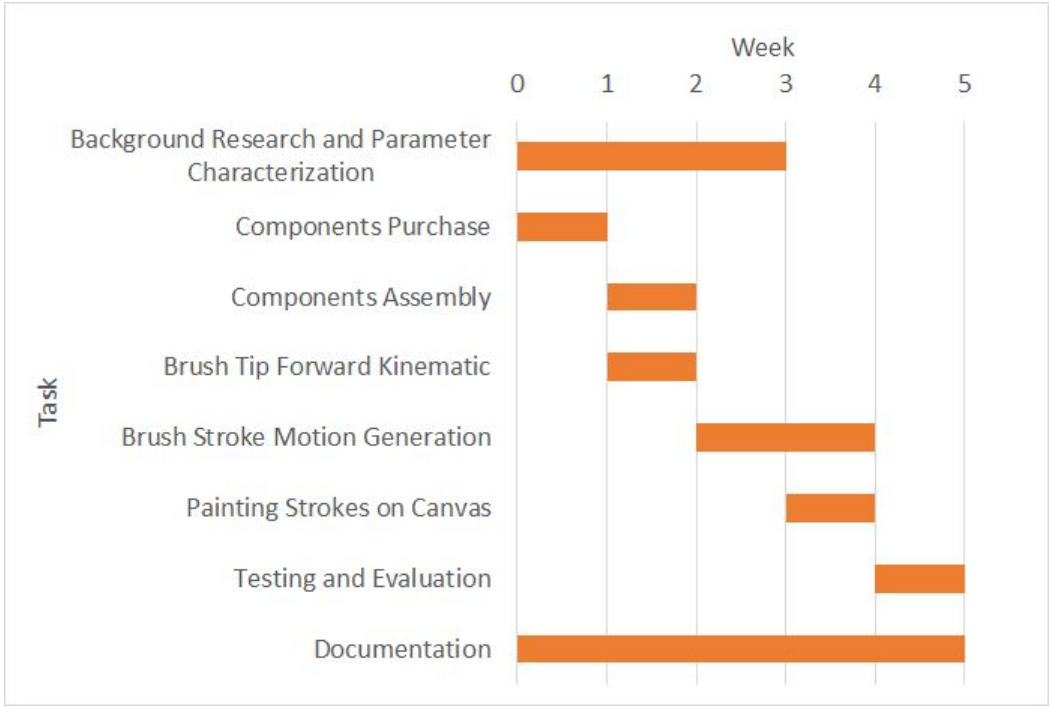


Future Directions

- Custom arm end effectors
  - allow for hot-swapping paintbrushes quickly and easily without measuring distances
- Controlling the arm to replenish paint on the brush instead of manually adding paint to the paintbrush
  - Different strokes can be created based on the amount of paint and wetness of the brush, instruct the arm to pick up more or less paint
- Additional sensors could make the arm aware of the relative position of the canvas so the canvas location must be fixed

Project Milestone and Goals

- Make a robot painter capable of drawing real paint strokes on the canvas.
- Project milestone:
  (5 Weeks)



Q&A