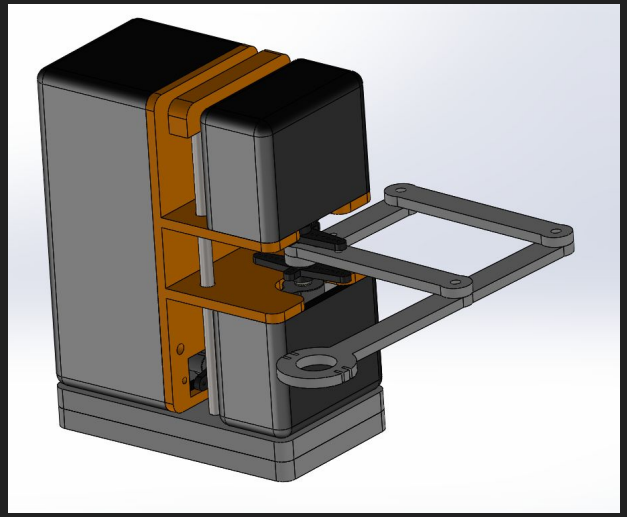
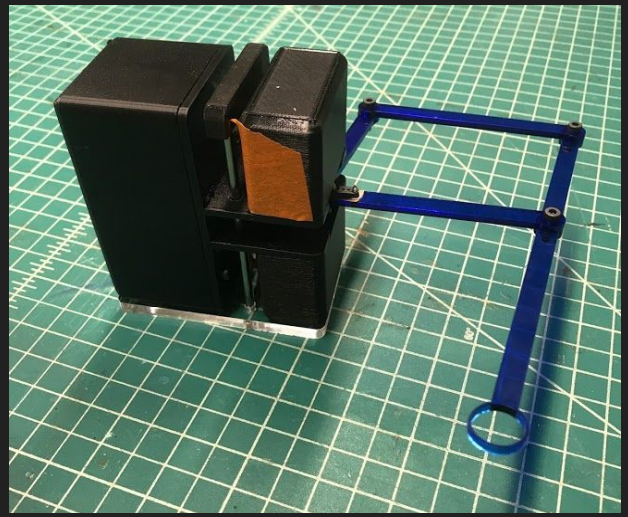
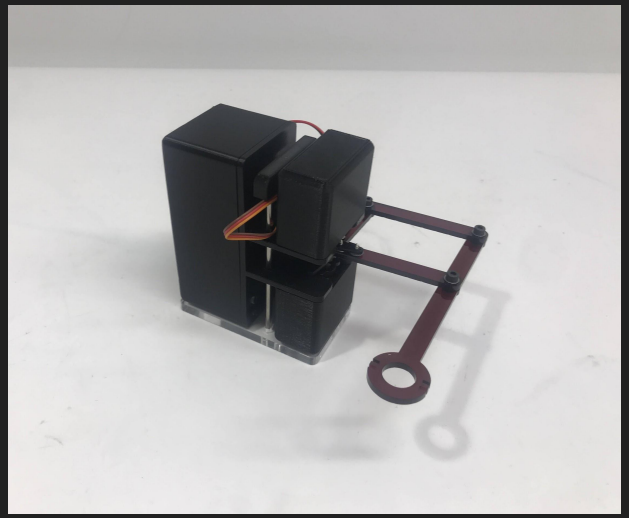
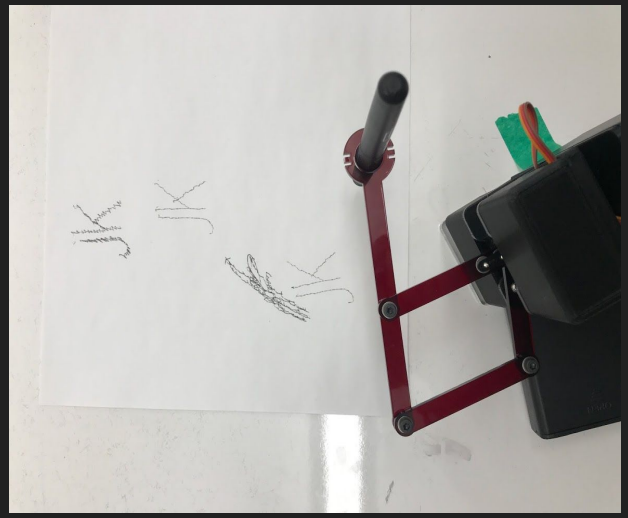
The beginning of a promising career in forgery

Max Kratzok & Jeremy Kanovsky









Inverse Kinematics

Links are stored with origin, length, and angle

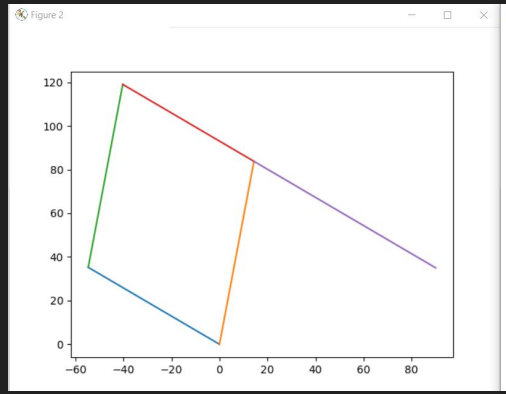
The tip of each link can be found using tip()

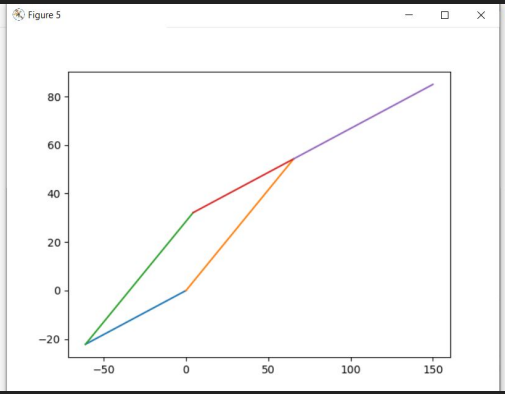
Solve system of equations to find angles

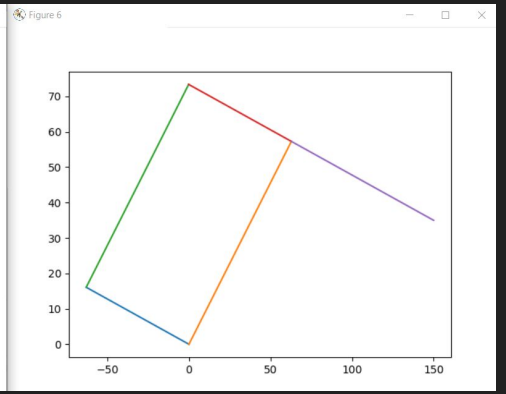
Return sets of angles

```
self.theta0 = sym.Symbol('theta0', real=True)
self.theta1 = sym.Symbol('theta1', real=True)
self.theta2 = sym.Symbol('theta2', real=True)
self.theta3 = sym.Symbol('theta3', real=True)

self.x2 = sym.Symbol('x2', real=True)
self.y2 = sym.Symbol('y2', real=True)
self.x3 = sym.Symbol('x3', real=True)
self.y3 = sym.Symbol('y3', real=True)

self.e1 = sym.Eq(self.x3, self.x2 + self.links[2].length * sym.cos(self.theta2))
self.e2 = sym.Eq(self.y3, self.y2 + self.links[2].length * sym.sin(self.theta2))
self.e3 = sym.Eq(self.x3, self.links[1].length * sym.cos(self.theta1) - self.links[3].length * sym.cos(self.theta3))
self.e4 = sym.Eq(self.y3, self.links[1].length * sym.sin(self.theta1) - self.links[3].length * sym.sin(self.theta3))
self.e5 = sym.Eq(self.x2, self.links[0].length * sym.cos(self.theta0))
self.e6 = sym.Eq(self.y2, self.links[0].length * sym.sin(self.theta0))
```

```
def getAngles(self, x4, y4):
    e7 = sym.Eq(x4, self.links[1].length * sym.cos(self.theta1) + self.links[4].length * sym.cos(self.theta3))
    e8 = sym.Eq(y4, self.links[1].length * sym.sin(self.theta1) + self.links[4].length * sym.sin(self.theta3))

    solution = sym.nsolve([self.e1, self.e2, self.e3, self.e4, self.e5, self.e6, e7, e8],
                          [self.theta0, self.theta1, self.theta2, self.theta3, self.x2, self.y2, self.x3, self.y3],
                          [math.radians(self.links[0].angle), math.radians(self.links[1].angle), math.radians(self.links[2].angle),
                           math.radians(self.links[3].angle), self.links[2].origin[0], self.links[2].origin[1], self.links[3].origin[0],
                           self.links[3].origin[1]])

    self.links[0].angle = math.degrees(solution[0])
    self.links[1].angle = math.degrees(solution[1])
    self.links[2].angle = math.degrees(solution[2])
    self.links[3].angle = math.degrees(solution[3])
    self.links[2].origin = [solution[4], solution[5]]
    self.links[3].origin = [solution[6], solution[7]]
    self.links[4].angle = self.links[3].angle
    self.links[4].origin = self.links[3].tip()

    return [math.degrees(solution[0]), math.degrees(solution[1])]
```







Pather

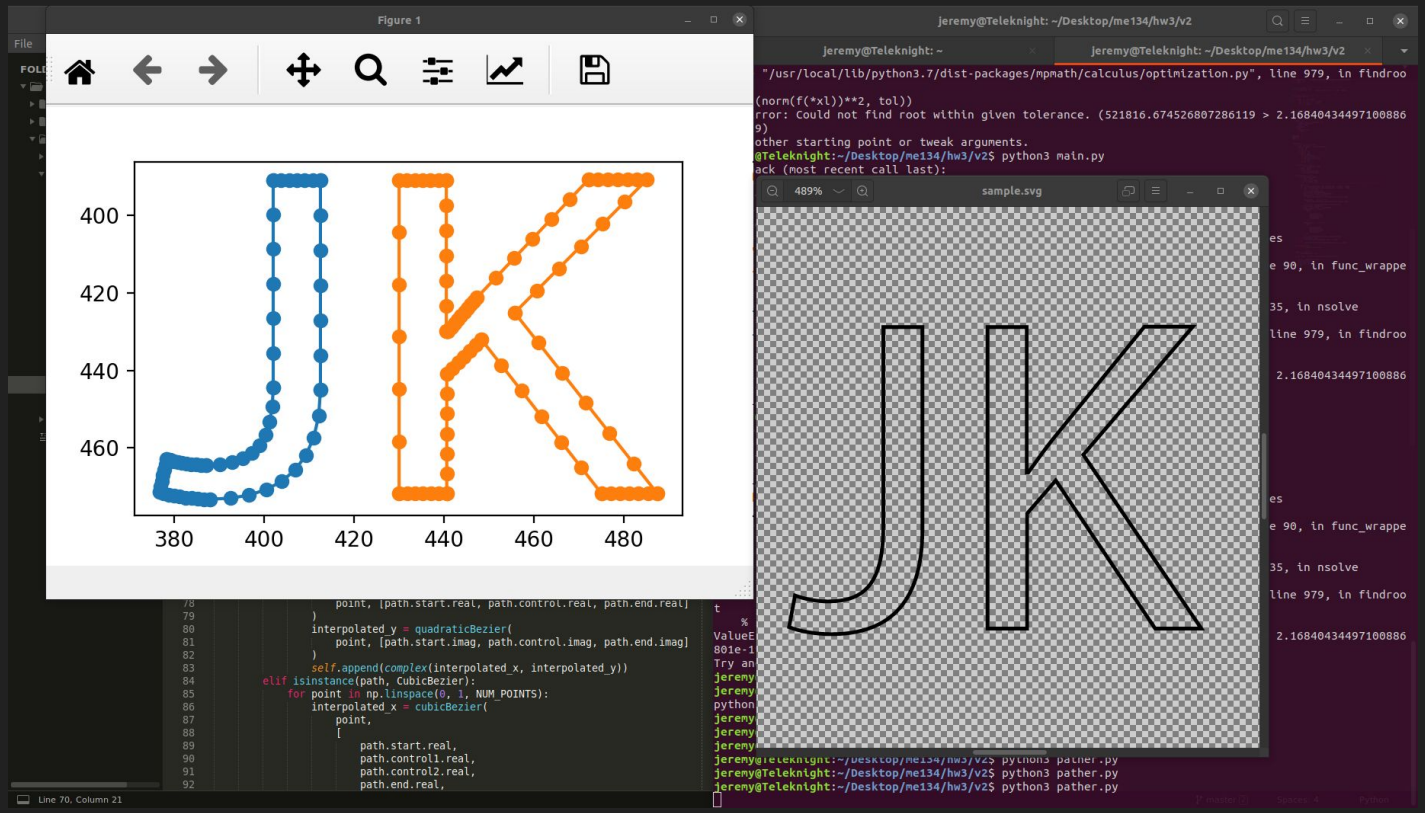
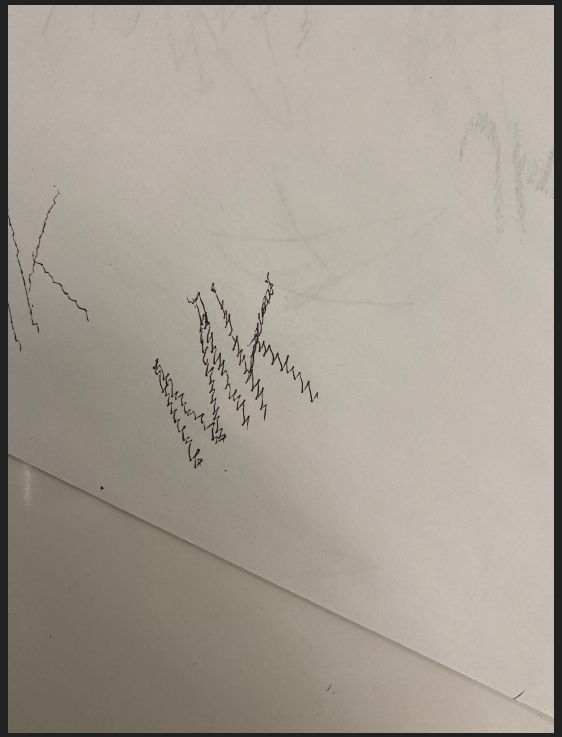


Figure 1
400
420
440
460
380
400
420
440
460
480
sample.svg
489%
jeremy@Teleknight: ~/Desktop/me134/hw3/v2
jeremy@Teleknight: ~
jeremy@Teleknight: ~/Desktop/me134/hw3/v2
"/usr/local/lib/python3.7/dist-packages/mpmath/calculus/optimization.py", line 979, in findroo
(norm(f(*xl))**2, tol))
rror: Could not find root within given tolerance. (521816.674526807286119 > 2.16840434497100886
9)
other starting point or tweak arguments.
@Teleknight:~/Desktop/me134/hw3/v2$ python3 main.py
ack (most recent call last):
jeremy@Teleknight:~/Desktop/me134/hw3/v2$ python3 pather.py
jeremy@Teleknight:~/Desktop/me134/hw3/v2$ python3 pather.py
jeremy@Teleknight:~/Desktop/me134/hw3/v2$ python3 pather.py
Line 70, Column 21

Command Line Arguments

```
usage: main.py [-h] [-i] input_file [-o] output_file [-s] style_factor [-a] arm [-dl]

Create and/or execute robot path

optional arguments:
-h, --help      show help message and exit
-i, --infile    input .svg or .pos file
-o, --outfile   output .pos file to save to
-s, --style     the desired robot arm style factor
-a, --arm       the arm set-up, 'short' or 'long'
-dl, --drawlive compiles .svg and runs path
```

Results



[End Slide]