EECE 487 - Assignment 6

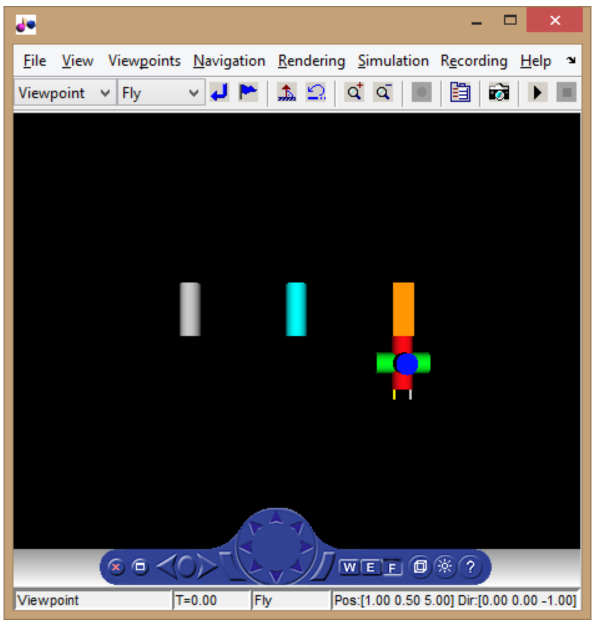
Dominik Boehi, Filip Juristovski, Nick Bojanowski

1)

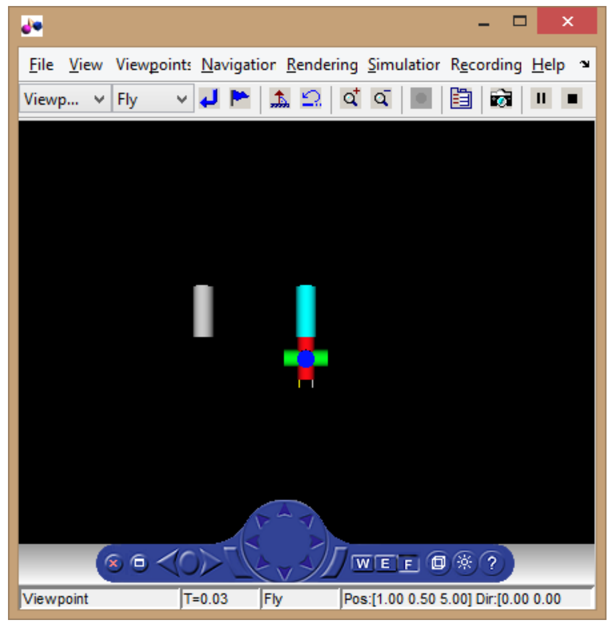
$$q = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T$$

$$q = [0, 90^o, 90^o, 0, 0, 0]^T$$

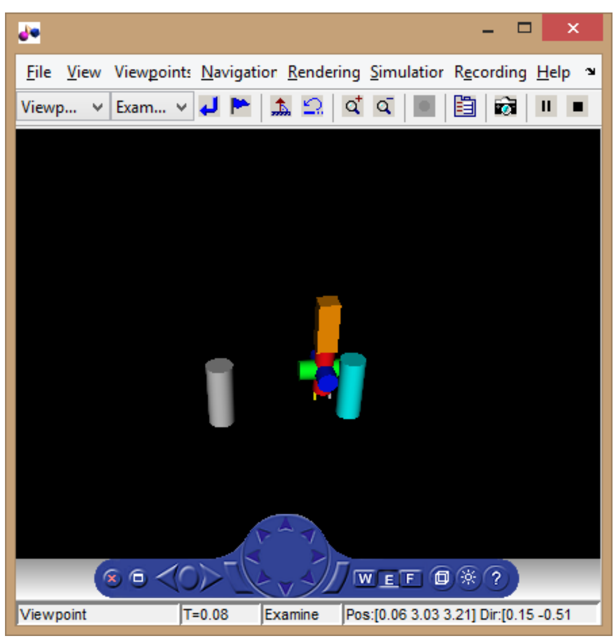
Before



After



After: Angled View



$$v = J(q)qDot$$

Gripper Velocity = J(q)(Joint Rates)

End Effector:

$$\{0\ 0\ 0\} + \{0, 0, (l_6+l_3)\} + \{l_1, 0, 0\} + \{0, l_2, 0\} - \{0, 0, l_3\} - \{0, 0, l_6\}$$

O_6 Coords = {1, 1, 0} with Orientation [1, -1, -1] = [i, -j, -k]

2) Matlab Script for Calculating q and qDot:

```
T = 1;
CalcPoints = 0:0.02:T;

steps = size(CalcPoints,2);

EndPos = [1;-0.5;0.5];
EndK = [0;0;-1];
EndJ = [1;0;0];

EndI = cross(EndJ, EndK);

EndC6 = [EndI EndJ EndK];

o6 = [1;1;0];

C6 = [1  0  0;
      0 -1  0;
      0  0 -1];

rotM = EndC6' * C6;

% Calculate gripper rotation
rotV = vrrotmat2vec(rotM);

angle = rotV(4);
axis = rotV(1:3)';

angular_v = sin(CalcPoints.*(pi/T));
```

```
angular_v = angular_v .* pi*angle/(2*T);

ang_v_vectors = axis * angular_v;

path = EndPos - o6;

d = norm(path);

% Calculate velocity vectors
v = sin(CalcPoints.*(pi/T));
v = v .* pi*d/(2*T);

v_vector = path/d;

if d == 0
     v_vector = zeros(3,1);
end

v_vectors = v_vector * v;

qStart = [0; -pi/2; 0; 0; 0; 0];

q = zeros(size(CalcPoints,2)+1, 6);
qDot = zeros(size(CalcPoints,2), 6);
qDotDot = zeros(size(CalcPoints,2), 6);

q(1,:) = qStart;

for i = 1:steps
     %Calculate velocity
     jacobian = ScaraJacobian(q(i,:)');

     if cond(jacobian) > 100
     disp('Singularity detected!');
     return
     end

     invJ = pinv(jacobian);
     qDot(i,:) = invJ*[v_vectors(:,i);ang_v_vectors(:,i)];
```

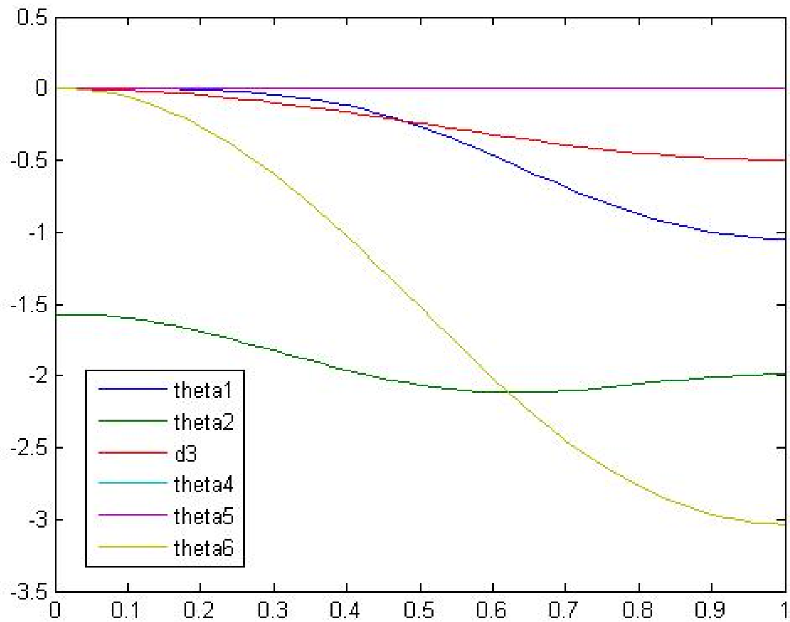
```
    %qDotDot(i,:) = invJ*(

    %Calculate new position
    q(i+1,:) = qDot(i,:)*0.02 + q(i,:);
end
q = q(1:end-1,:);

figure;
plot(CalcPoints, q);
legend('theta1', 'theta2', 'd3', 'theta4', 'theta5',
'theta6');
figure;
plot(CalcPoints, qDot);
legend('omega1', 'omega2', 'v3', 'omega4', 'omega5',
'omega6');
```

The code above takes a hardcoded end point and orientation for the gripper, in this case `EndPos = [1;-0.5;0.5]; EndK = [0;0;-1]; EndJ = [1;0;0];`, and a hardcoded 20ms time step, and a hardcoded runtime T (of 1 second), and creates a matrix q containing rows of joint variables. qDot is also created containing rows of joint velocities. Creation of these matrices are not done using inverse kinematics.

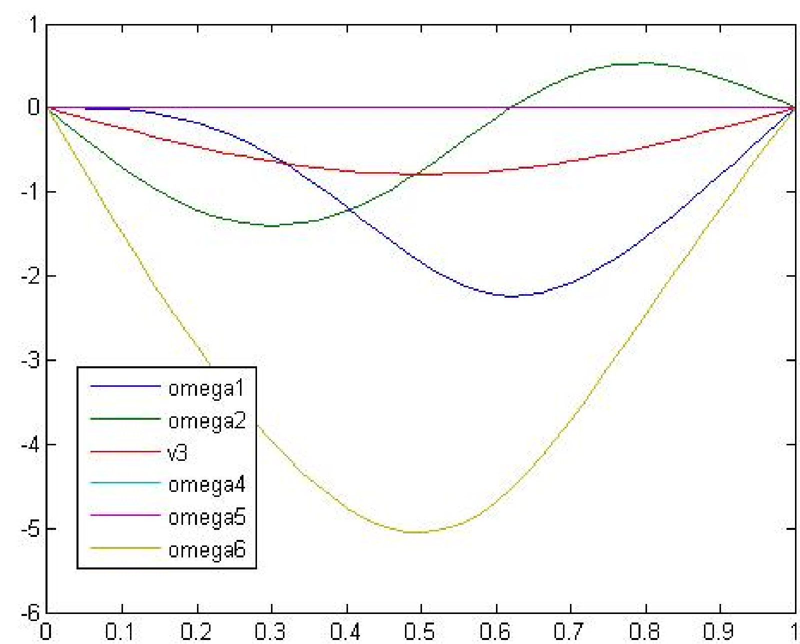
The condition number of the Jacobian is monitored, and trajectory is stopped if a singularity is approached. This was tested by starting the trajectory in a singularity and observing the error message.

The next page contains graphs of the joint variables and velocities over time.

q: Joint variables over Time



q: Joint Velocity over Time



The script was implemented in simulink and the rows of the joint variables in q were stepped through to simulate animation.

Simulink Model