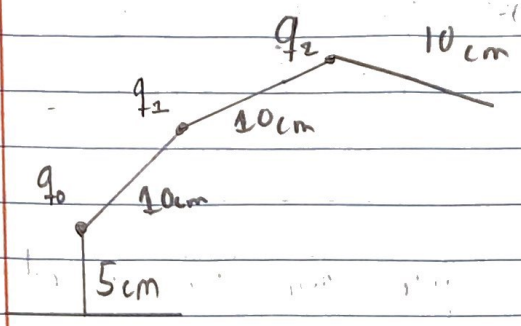
CSE 276C HOMEWORK 1

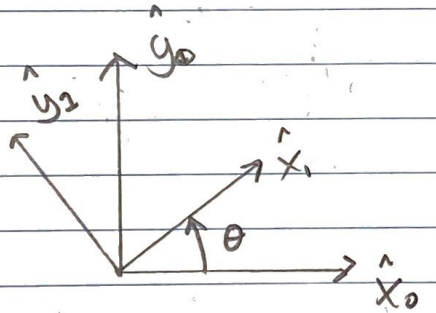
1



3 degrees of freedom

a) "Forward Kinematics. Derive $\vec{pr}(q_0, q_1, q_2)$ and $\dot{x} = T\dot{q}$"

If we have two coordinate frames:



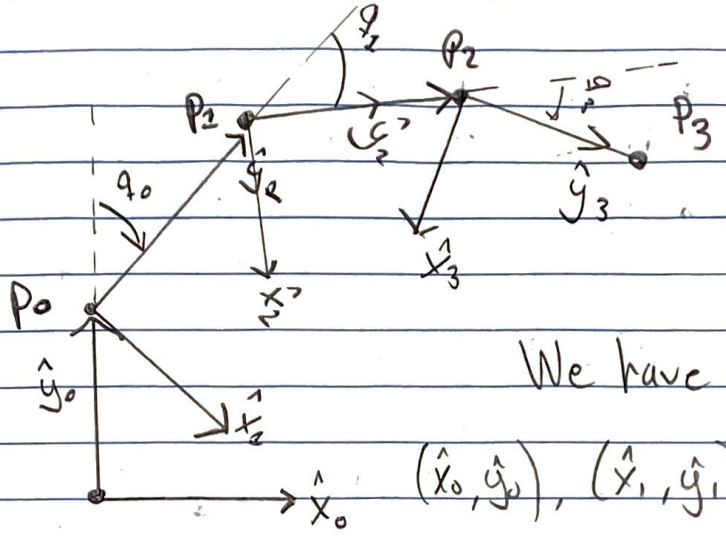
We define positive rotation counter clockwise.

To get from frame 1 to frame 0 we need to multiply with the rotation about the z-axis

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

I will use this knowledge on this problem

1 a)



We have four coordinate frames:

$(\hat{x}_0, \hat{y}_0)$, $(\hat{x}_1, \hat{y}_1)$, $(\hat{x}_2, \hat{y}_2)$, $(\hat{x}_3, \hat{y}_3)$

Frame 1 is rotated $-q_0$ degrees about the z-axis from frame 0. So the rotation matrix becomes,

$$R_1^0 = \begin{bmatrix} \cos(-q_0) & -\sin(q_0) & 0 \\ \sin(-q_0) & \cos(q_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(q_0) & +\sin(q_0) & 0 \\ -\sin(q_0) & \cos(q_0) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Frame 2 is rotated $-q_1$ from 1 and frame 3 rotated $-q_2$ from frame 2 and we get

$$R_1^2 = \begin{bmatrix} \cos(q_1) & \sin(q_1) & 0 \\ -\sin(q_1) & \cos(q_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_3^2 = \begin{bmatrix} \cos(q_2) & \sin(q_2) & 0 \\ -\sin(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

1 a) If you have a vector in b frame and want it in a frame you rotated.

$${}^{a}r = R_b^a \, {}^{b}r.$$

To find the end point in 0 frame we need to add:

$$\vec{P_r} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}_0 + \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix}_1 + \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix}_2 + \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix}_3$$

We apply the rotation matrices:

$$\vec{P_r} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}_0 + R_1^0 \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix}_1 + R_1^0 R_2^1 \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix}_2 + R_1^0 R_2^1 R_3^2 \begin{bmatrix} 0 \\ 10 \\ 0 \end{bmatrix}_3$$

$$= \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix}_0 + \begin{bmatrix} 10\sin(q_0) \\ 10\cos(q_0) \\ 0 \end{bmatrix}_0 + \begin{bmatrix} 10\sin(q_0+q_1) \\ 10\cos(q_0+q_1) \\ 0 \end{bmatrix}_0 + \begin{bmatrix} 10\sin(q_0+q_1+q_2) \\ 10\cos(q_0+q_1+q_2) \\ 0 \end{bmatrix}_0$$

since all are in 0-frame we can add and get:

$$\vec{P_r} = \begin{bmatrix} 10\sin(q_0) + 10\sin(q_0+q_1) + 10\sin(q_0+q_1+q_2) \\ 5 + 10\cos(q_0) + 10\cos(q_0+q_1) + 10\cos(q_0+q_1+q_2) \\ 0 \end{bmatrix}_0$$

1 a) Since we are in 2D we drop the last row (z) and get

$$\vec{p_r} = \begin{bmatrix} x \\ y \end{bmatrix}$$

To find motion model $\dot{x} = T\dot{q}$

we need to find the Jacobian for $\vec{p_r}$.

Jacobian $\begin{bmatrix} \frac{\partial x}{\partial q_0} & \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} \\ \frac{\partial y}{\partial q_0} & \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} \end{bmatrix}$

What we use here is the chain rule

$f(x(t))$ the motion is $\frac{\partial f}{\partial x} \cdot \dot{x}$

So $\frac{d}{dt} \vec{P_r}(q_0, q_1, q_2) = \underbrace{\frac{\partial \vec{p}}{\partial \vec{q}}}_{\text{Jacobian} = T} \dot{q}$

1 a) I solve for Jacobian in Matlab and get

$$\dot{X} = \bar{J}\,\dot{q} = \begin{bmatrix} \frac{\partial x}{\partial q_0} & \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} \\ \frac{\partial y}{\partial q_0} & \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} \end{bmatrix} \begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

$$\frac{\partial x}{\partial q_0} = 10\cos(q_0+q_1+q_2) + 10\cos(q_0+q_1) + 10\cos(q_0)$$

$$\frac{\partial x}{\partial q_1} = 10\cos(q_0+q_1+q_2) + 10\cos(q_0+q_1)$$

$$\frac{\partial x}{\partial q_2} = 10\cos(q_0+q_1+q_2)$$

$$\frac{\partial y}{\partial q_0} = -10\sin(q_0+q_1+q_2) - 10\sin(q_0+q_1) - 10\sin(q_0)$$

$$\frac{\partial y}{\partial q_1} = -10\sin(q_0+q_1+q_2) - 10\sin(q_0+q_1)$$

$$\frac{\partial y}{\partial q_2} = -10\sin(q_0+q_1+q_2)$$

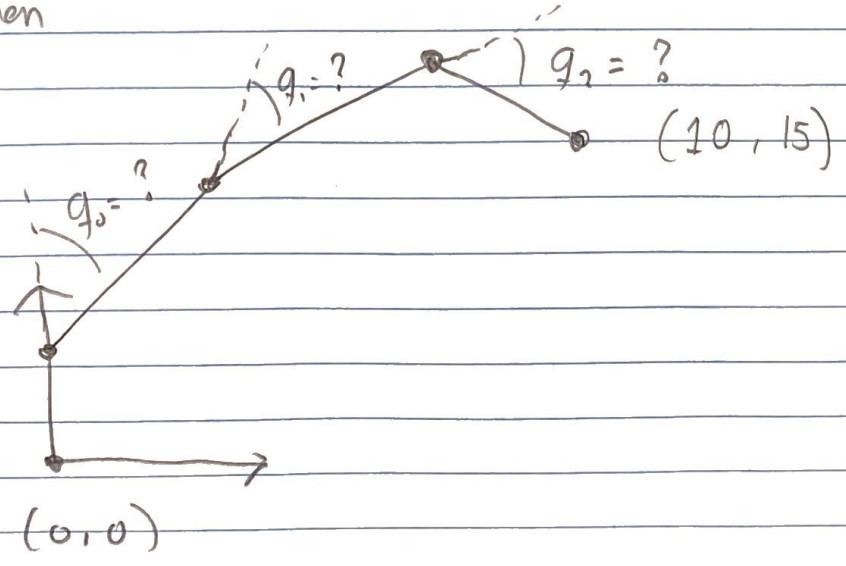
1 a) That means that $\dot{X} =$

$$\dot{X} = \begin{bmatrix} \frac{\partial x}{\partial q_0} \cdot \dot{q}_0 + \frac{\partial x}{\partial q_1} \dot{q}_1 + \frac{\partial x}{\partial q_2} \dot{q}_2 \\ \frac{\partial y}{\partial q_0} \cdot \dot{q}_0 + \frac{\partial y}{\partial q_1} \dot{q}_1 + \frac{\partial x}{\partial q_2} \dot{q}_2 \end{bmatrix}$$

b) "Inverse Kinematics. Determine joint angle conf. for position at the end (10, 15) "

This means we want to know q_0, q_1, q_2

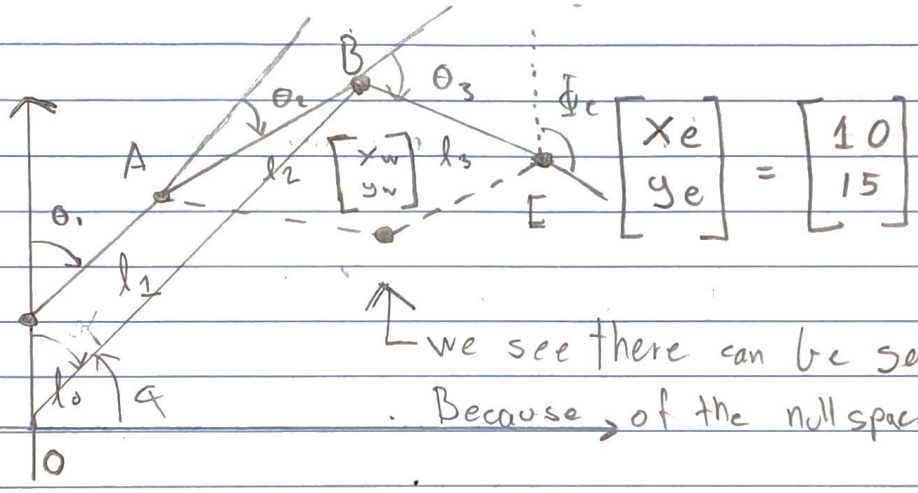
When



```
clc; clear;
syms q0 q1 q2 q0_dot q1_dot q2_dot

% Rotation matices
R_01 = [cos(q0) sin(q0) 0;
        -sin(q0) cos(q0) 0;
          0       0      1];

R_12 = [cos(q1) sin(q1) 0;
        -sin(q1) cos(q1) 0;
          0       0      1];
R_23 = [cos(q2) sin(q2) 0;
        -sin(q2) cos(q2) 0;
          0       0      1];

% Length between joints
l0 = [0 5 0].';
l1 = [0 10 0].';
l2 = [0 10 0].';
l3 = [0 10 0].';

r1 = R_01*l1;
r2 = R_01*R_12 * l2;
r3 = R_01*R_12*R_23 * l3;

% Finding vector to end point
p_r = simplify(l0 + r1 + r2 + r3);

% Define the joint variables and joint velocities
joint_vars = [q0, q1, q2];
joint_velocities = [q0_dot, q1_dot, q2_dot];

% Calculate the Jacobian matrix
J_p = simplify(jacobian(p_r, joint_vars));
```

1 b) There is none unique solution to this problem since there is several ways for the robot to find the point.

We sketch the robot once more



we see there can be several solutions. Because of the null space

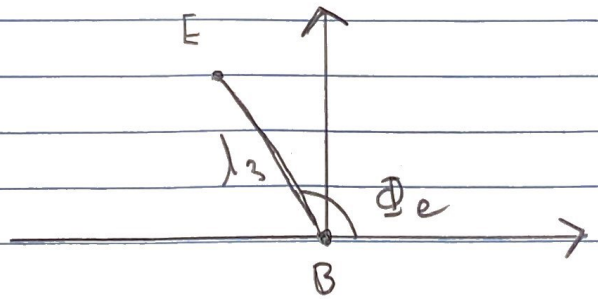
We start by finding the "wrist" $\begin{bmatrix} x_w \\ y_w \end{bmatrix}$

To get from the wrist to the end we need to go:

$$x_w + l_3 \cdot \cos(\Phi_e) = x_e$$
$$y_w + l_3 \sin(\Phi_e) = y_e$$

(*) This derivation comes from [1] "Planar Kinematics"

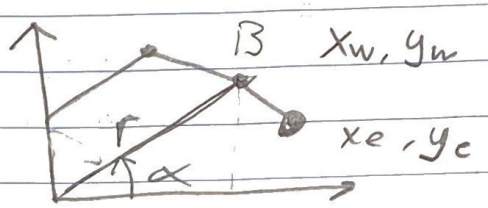
from this we can see it:



1 b) We can determine the "wrist" coordinates now from

B

$$\alpha = \tan^{-1}\left[\frac{y_w}{x_w}\right]$$

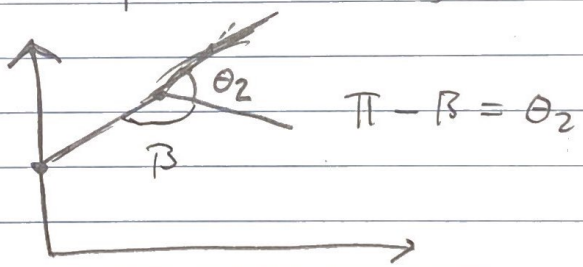


Now we use the law of cosine on

$$l_1^2 + l_2^2 + 2 l_1 l_2 \cos\beta = r^2$$

$$r^2 = x_w^2 + y_w^2 \quad \text{(dist. from O to B)}$$

$$\Theta_2 = \pi - \beta$$



β can be found from the law of cosine:

$$\theta_2 = \pi - \beta = \pi - \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x_w^2 - y_w^2}{2 l_1 l_2}\right)$$

We use another triangle to find

$$r^2 + l_1^2 - 2 r l_1 \cos\gamma = l_2^2$$

$$\theta_1 = \alpha - \gamma = \tan^{-1}\left(\frac{y_w}{x_w}\right) - \cos^{-1}\left(\frac{x_w^2 + y_w^2 + l_1^2 - l_2^2}{2 l_1 \sqrt{x_w^2 + y_w^2}}\right)$$

1 (b) We then can find θ_3

$$\theta_3 = \Phi e - \theta_1 - \theta_2 .$$

So the equations we use in Matlab becomes:

$$\theta_1 = \tan^{-1}\left(\frac{y_w}{x_w}\right) - \cos^{-1}\left(\frac{l_1^2 + l_2^2 + x_w^2 + y_w^2}{2 l_2 \sqrt{x_w^2 + y_w^2}}\right) \quad (1)$$

$$\theta_2 = \pi - \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x_w^2 - y_w^2}{2 l_1 l_2}\right) \quad (2)$$

$$\theta_3 = \Phi_e - \theta_1 - \theta_2 \quad (3)$$

$$x_w = x_e - l_3 \cos \Phi_e$$
$$y_w = y_e - l_3 \sin \Phi_e$$

The only thing left is to choose Φe to satisfy the constraints. I tried with some values until all θ was $\pm 90°$

With $\Phi_e = 270° / \frac{\pi}{1.5}$ rad

We get:

$\theta_1 = -36°$

$\theta_2 = 82°$

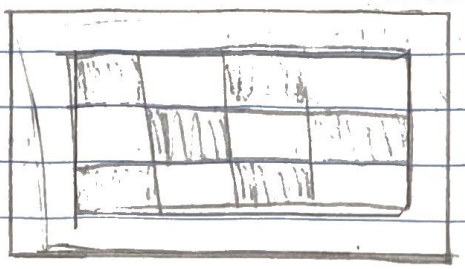
$\theta_3 = 74°$

And checking wit forward kin.

$$\vec{p}_r(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} 10.02 \\ 15.04 \end{bmatrix}$$

```
clc; clear;

% Length of arms
l1 = 10;
l2 = 10;
l3 = 10;

% Desired end position
x_e = 10;
y_e = 10; % I subtract 5 since the joint starts at (0,5) from the Origin

% Angle in radians
psi_e = pi/1.5;

% Formula to find inverse Kinematics
x_w = x_e - l3*cos(psi_e);
y_w = y_e - l3*sin(psi_e);

theta1 = atan(y_w/x_w) - acos((l1^2-l2^2+x_w^2+y_w^2)/(2*l1*sqrt(x_w^2+y_w^2)));
theta2 = pi - acos((l1^2+l2^2-x_w^2-y_w^2)/(2*l1*l2));
theta3 = psi_e - theta1 - theta2;

% Convert angles to degrees
theta1_deg = rad2deg(theta1);
theta2_deg = rad2deg(theta2);
theta3_deg = rad2deg(theta3);

% Display angles in degrees
disp(['theta1 (degrees): ', num2str(theta1_deg)]);
disp(['theta2 (degrees): ', num2str(theta2_deg)]);
disp(['theta3 (degrees): ', num2str(theta3_deg)]);
```

2 Camera Calibration



Sketch of the board

To determine 3D world $\underline{X}$ from ref. $\underline{u}$ we need $[\underline{\underline{R}} \mid \underline{t}]$ maps world to camera. $\underline{\underline{A}}$ intrinsc matrix project onto 2D plane. $\underline{u} = \underline{\underline{A}}\,[\underline{\underline{R}} \mid \underline{t}]\,\underline{X}$

In homogeneous coordinates

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- f_x, f_y focal length
- c_x, c_y img center
- s scale factor

Goal is to find values of intrinsic and extrinsic matrices for a set of calibration images.

2.1 "Plane - to - Plane Homography"

"The file contains 2D pixels for corner from 9 img.

2D homography H is 3×3

$\underline{u} = \underline{\underline{H}}\,\underline{p}$ $\quad \underline{p} = [X, Y, 1]$ - checkerboard coordinates

$\underline{u} = [u, v, 1]$ - image coordinates

2.1 a) "Let $\underline{h}$ be flattend 9-vector $\underline{\underline{H}}$, given $p_i \rightarrow u_i$ rearrange $u_i = Hp_i$ into a system of linear eq. $M_i h = 0$."

We have: $\underline{p} = [x \; y \; 1]^T$

$\underline{u} = [u \; v \; 1]^T$

$$\underline{\underline{H}} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

If we set up the equation:

$$\underline{u} = \underline{\underline{H}}\,\underline{p}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} - \begin{bmatrix} x h_{11} + y h_{12} + h_{13} \\ x h_{21} + y h_{22} + h_{23} \\ x h_{31} + y h_{32} + h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

We know $\underline{h}$ is like this:

$$\underline{h} = [h_{11} \; h_{12} \; h_{13} \; h_{21} \; h_{22} \; h_{23} \; h_{31} \; h_{32} \; h_{33}]^T$$

2.1 a) We have three equations:

$$0 = h_{11}x + h_{12}y + h_{13} - u \cdot 1 \quad (1)$$

$$0 = h_{21}x + h_{22}y + h_{23} - v \cdot 1 \quad (2)$$

$$1 = h_{31}x + h_{32}y + h_{33} \quad (3)$$

The (3) equals 1 so we substitute into (1), (2)

$$0 = h_{11}x + h_{12}y + h_{13} - u\underbrace{(h_{31}x + h_{32}y + h_{33})}_{=1}$$

$$0 = h_{21}x + h_{22}y + h_{23} - v\underbrace{(h_{31}x + h_{32}y + h_{33})}_{=1}$$

We can now construct:

$$\underline{\underline{M_i}} = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xu & yu & u \\ 0 & 0 & 0 & -x & -y & -1 & xv & yv & v \end{bmatrix}$$

Now:

$$u = Hp \rightarrow \quad M_i h = 0$$

2.1 a) i) "How many linear eq. to get from one point correspond?"

We get two linear equations from one point

ii) "How many do we need to fully define H?"

H has 9 element so we needs 9 equations to fully define H

iii) "Given number of ref. points is images homography over/under/fully-defined!"

The systems homography is ove defined due to the fact we have more points and ref. points then we need to slve the equations.

b) "Construct M by stacking M_i for each ref point within one image. Solve LLS $Mh = 0$. Report the reprojection error"

See Matlab pdf.

The total Eclidian distance = 162.8

2.1 b) I start by loading imgpoints

u, v are 432 long so I mak. X, Y to be 432 long to match

Then for each image for each point I make the M_i and stack to M.

Then I use svd(M) to find the H corresponding to that Image. Resulting in 9 different matrices one for each image. See next page and Matlab code.

Finally I computed $p = [x, y, 1]$ with $Hp = [u', v', w]$ and divided $u^* = \left[\frac{u'}{w}, \frac{v'}{w}\right]$ and took $\sum |u - u^*|$ (sum of error)

Sum of Euclidian distance = 162.8

(for all points of all images

THE H-MATRIX

2.1 G)

$$H_1 = \begin{bmatrix} 0.0054 & -0.0017 & -0.6205 \\ -3.48 \cdot 10^{-4} & 0.0014 & 0.7841 \\ 6.56 \cdot 10^{-7} & -6.97 \cdot 10^{-6} & 0.0044 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 0.0051 & -8 \cdot 10^{-4} & 0.7177 \\ -4.97 \cdot 10^{-5} & 0.0014 & 0.6963 \\ 1.78 \cdot 10^{-6} & -6.0 \cdot 10^{-6} & 0.0039 \end{bmatrix}$$

$$H_3 = \begin{bmatrix} -0.0063 & -2.0 \cdot 10^{-4} & -0.5050 \\ -1.48 \cdot 10^{-4} & -0.0023 & -0.8631 \\ -3.87 \cdot 10^{-6} & 5.84 \cdot 10^{-6} & -0.0043 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} -0.0063 & -8.22 \cdot 10^{-4} & -0.7498 \\ -3.6 \cdot 10^{-4} & -0.0026 & -0.6616 \\ -4.68 \cdot 10^{-6} & 5.3 \cdot 10^{-6} & -0.0038 \end{bmatrix}$$

2.16)

$$H_5 = \begin{bmatrix} -0.0051 & -6.62 \cdot 10^{-4} & -0.5600 \\ -3.15 \cdot 10^{-4} & -0.0021 & -0.8284 \\ -3,91 \cdot 10^{-6} & 4,32 \cdot 10^{-6} & -0.0034 \end{bmatrix}$$

$$H_6 = \begin{bmatrix} -0.0057 & -0,0012 & -0.6426 \\ -5.33 \cdot 10^{-4} & -0.0024 & -0.7661 \\ -5.08 \cdot 10^{-6} & 4.30 \cdot 10^{-6} & -0,0037 \end{bmatrix}$$

$$H_7 = \begin{bmatrix} -0.0057 & -0.0012 & -0.3996 \\ -5.76 \cdot 10^{-4} & -0,0024 & -0.4166 \\ -5.20 \cdot 10^{-6} & 4.35 \cdot 10^{-6} & -0.0037 \end{bmatrix}$$

$$H_8 = \begin{bmatrix} -0.0049 & -0,0011 & -0.7806 \\ 9.21 \cdot 10^{-5} & -0.0023 & -0.6250 \\ -3.78 \cdot 10^{-6} & 3,59 \cdot 10^{-6} & -0.0033 \end{bmatrix}$$

$$H_9 = \begin{bmatrix} -0,0045 & -7.3 \cdot 10^{-4} & -0.7539 \\ -3,749 \cdot 10^{-4} & -0,0015 & -0.6570 \\ -3.68 \cdot 10^{-6} & 4.11 \cdot 10^{-6} & -0,0031 \end{bmatrix}$$

```
clc; clear;

% Load the 2D image coordinates (u, v) from the file
data = dlmread('imgpoints.txt');

% Constants
numX = 8;
numY = 6;
numPointsPerImage = 48;
numImages = 9;
squareSize = 25;

% Extract the 2D image coordinates
u = data(:, 1);
v = data(:, 2);

% Generate the 3D grid coordinates (X, Y)
X = zeros(numX * numY, 1);
Y = zeros(numX * numY, 1);

% Create the grid pattern and repeat for each image
for img = 1:numImages
    for i = 1:numY
        for j = 1:numX
            idx = (img - 1) * numX * numY + (i - 1) * numX + j;
            X(idx) = (j - 1) * squareSize;
            Y(idx) = (i - 1) * squareSize;
        end
    end
end

% Initialize the homography matrices
H = [];

for img = 1:numImages
    % Initialize the matrix M by stacking M_i for each reference point
    M = [];

    for point = (numPointsPerImage * (img-1) + 1) : numPointsPerImage * img
        u_i = u(point);
        v_i = v(point);
        x_i = X(point);
        y_i = Y(point);

        M_i = [-x_i, -y_i, -1, 0, 0, 0, x_i * u_i, y_i * u_i, u_i;
               0, 0, 0, -x_i, -y_i, -1, x_i * v_i, y_i * v_i, v_i];
        M = [M; M_i];
    end

    % Solve the equation M * h = 0
    [~, ~, V] = svd(M);
    h = V(:, end);
    H = [H h];
end
```

```
% Calculate the reprojection error
euclidean_distance = 0;
for img = 1:numImages
    % Extract the H - homography matrix for each image
    H_img = reshape(H(:, img), 3, 3)';

    for point = (numPointsPerImage * (img-1) + 1) : numPointsPerImage * img
        p = [X(point); Y(point); 1];
        u_calc = H_img * p;
        w = u_calc(3);

        u_guess = [u_calc(1)/w; u_calc(2)/w];
        u_correct = [u(point); v(point)];

        % Calculate the Euclidean distance
        euclidean_distance = euclidean_distance + norm(u_correct-u_guess);
    end
end

fprintf('Total reprojection error all images: %.2f\n', euclidean_distance);

% Save the variables to MAT files
save('image_coordinates.mat', 'u', 'v');
save('world_coordinates.mat', 'X', 'Y');
save('homography_matrix.mat', 'H');
save('constants.mat','numImages','numPointsPerImage');
```

2.2 "Instrinsic Matrix. A related to homography

(1) $h_1^T A^{-T} A^{-1} h_2 = 0$ h_i: ith column of H

(2) $h_1^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2$

$B = A^{-T} A^{-1}$ $b = [b_{11}\ b_{12}\ b_{13}\ b_{22}\ b_{23}\ b_{33}]$ symetric"

a) "Given H_i use constraints to form $L_i b = 0$"

Since B is symetic it looks like this:

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix}$$

We know $B = A^{-T} A^{-1}$ so we can insert into (1)(2)

(1) $h_1^T A^{-T} A^{-1} h_2 - h_1^T B h_2 = 0$

(2) $h_1^T A^{-T} A^{-1} h_1 = h_1^T B h_1 = h_2^T B h_2$

Given $H_i = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$

h_1 h_2 h_3

2.2 a)

(1) $$\begin{bmatrix} h_{11} & h_{21} & h_{31} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \begin{bmatrix} h_{12} \\ h_{22} \\ h_{32} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$h_{11}(h_{12}b_{11} + h_{22}b_{12} + h_{32}b_{13}) +$$
$$h_{21}(h_{12}b_{12} + h_{22}b_{22} + h_{32}b_{23}) +$$
$$h_{31}(h_{12}b_{13} + h_{22}b_{23} + h_{32}b_{33}) = 0$$

$$\underline{h_{11}h_{12}b_{11}} + \underline{h_{11}h_{22}b_{12}} + \underline{h_{11}h_{32}b_{13}} +$$
$$\underline{h_{21}h_{12}b_{12}} + \underline{h_{21}h_{22}b_{22}} + \underline{h_{21}h_{32}b_{23}} +$$
$$\underline{h_{31}h_{12}b_{13}} + \underline{h_{31}h_{22}b_{23}} + \underline{h_{31}h_{32}b_{33}} = 0$$

$$(h_{11}h_{12})\cdot b_{11} + (h_{11}h_{22} + h_{21}h_{12})b_{12} + (h_{11}h_{32} + h_{31}h_{12})b_{13} +$$
$$(h_{21}h_{22})b_{22} + (h_{21}h_{32} + h_{31}h_{22})b_{23} + (h_{31}h_{32})b_{33} = 0$$

$$\begin{bmatrix} h_{11}h_{12} \\ h_{11}h_{22} + h_{21}h_{12} \\ h_{11}h_{32} + h_{31}h_{12} \\ h_{21}h_{22} \\ h_{21}h_{32} + h_{31}h_{22} \\ h_{31}h_{32} \end{bmatrix}^T \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{23} \\ b_{33} \end{bmatrix} = 0 \quad \text{(I)}$$

2.2 a) Now we look at eq. (2) $B = A^{-T} A^{-1}$

$$h_2^T A^{-T} A^{-1} h_1 = h_2^T A^{-T} A^{-1} h_2 \quad (2)$$

$$h_1^T B h_1 = h_2^T B h_2$$

$$\begin{bmatrix} h_{11} & h_{21} & h_{31} \end{bmatrix} B \begin{bmatrix} h_{11} \\ h_{21} \\ h_{31} \end{bmatrix} = \begin{bmatrix} h_{12} & h_{22} & h_{32} \end{bmatrix} B \begin{bmatrix} h_{12} \\ h_{22} \\ h_{32} \end{bmatrix}$$

$$h_{11}(h_{11}b_{11} + h_{21}b_{12} + h_{31}b_{13}) + h_{12}(h_{12}b_{11} + h_{22}b_{12} + h_{32}b_{13}) +$$

$$h_{21}(h_{11}b_{12} + h_{21}b_{22} + h_{31}b_{23}) + h_{22}(h_{12}b_{12} + h_{22}b_{22} + h_{32}b_{23}) +$$

$$h_{31}(h_{11}b_{13} + h_{21}b_{23} + h_{31}b_{33}) \quad h_{32}(h_{12}b_{13} + h_{22}b_{23} + h_{32}b_{33})$$

$$(h_{11}^2 - h_{12}^2)b_{11} + (h_{11}h_{21} + h_{21}h_{11} - h_{12}h_{22} - h_{12}h_{22})b_{12} +$$

$$(h_{11}h_{31} + h_{11}h_{31} - h_{12}h_{32} - h_{12}h_{32})b_{13} + (h_{21}^2 - h_{22}^2)b_{22} +$$

$$(h_{21}h_{31} + h_{21}h_{31} - h_{22}h_{32} - h_{22}h_{32})b_{23} + (h_{31}^2 - h_{32}^2)b_{33}$$

$$\begin{bmatrix} h_{11}^2 - h_{12}^2 \\ 2h_{11}h_{21} - 2h_{12}h_{22} \\ 2h_{11}h_{31} - 2h_{12}h_{32} \\ h_{21}^2 - h_{22}^2 \\ 2h_{21}h_{31} - 2h_{22}h_{32} \\ h_{31}^2 - h_{32}^2 \end{bmatrix}^T \cdot \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{22} \\ b_{23} \\ b_{33} \end{bmatrix} = 0 \quad (II)$$

2.2 a) We put I and II together:

$$\begin{bmatrix} h_{11}h_{12} & h_{11}^2 - h_{12}^2 \\ h_{11}h_{22} + h_{21}h_{12} & 2h_{11}h_{21} - 2h_{12}h_{22} \\ h_{11}h_{32} + h_{31}h_{12} & 2h_{11}h_{31} - 2h_{12}h_{32} \\ h_{21}h_{22} & h_{21}^2 - h_{22}^2 \\ h_{21}h_{32} + h_{31}h_{22} & 2h_{21}h_{31} - 2h_{22}h_{32} \\ h_{31}h_{32} & h_{31}^2 - h_{32}^2 \end{bmatrix}^T \begin{bmatrix} G_{11} \\ G_{12} \\ G_{13} \\ G_{22} \\ G_{23} \\ G_{33} \end{bmatrix} = 0$$

L $\quad$ b

i) "How many linear eq. from one homography?"

We see from Lb we get 2 linear eq.

ii) "How many img. to forfill b?"

We need 6 eq. because b is 6 unknown. So 3 images.

iii) "Over/under/fully-defined"

Given 9 images this is over defined.

2.2 b) "Construct L by stacking L_i for each images. Solve $L_i b = 0$

I start by loading each H_i for picture i and making h_{i1}, h_{i2} ...

Then I construct L_i and stack them.

After using svd(L) I get the b

$$b = \begin{bmatrix} -1.6 \cdot 10^{-6} \\ 2.9 \cdot 10^{-7} \\ 6.1 \cdot 10^{-4} \\ -2.8 \cdot 10^{-6} \\ -1.2 \cdot 10^{-4} \\ -1 \end{bmatrix}$$

c) "Using eq. appendix B, calculate A"

In appendix B we can find several equations to find A.

2.2 c)

We now have matrix B

$$B = \begin{bmatrix} b_1 & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \quad \begin{aligned} b_1 &= -1.6 \cdot 10^{-6} & b_4 &= -2.8 \cdot 10^{-6} \\ b_2 &= 7.9 \cdot 10^{-7} & b_5 &= -1.2 \cdot 10^{-4} \\ b_3 &= 6.1 \cdot 10^{-4} & b_6 &= -1 \end{aligned}$$

The eq. from appendix B gives:

$$v_0 = (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^2) = 219$$

$$\lambda = B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})] / B_{11} = -0.696$$

$$\alpha = \sqrt{\lambda / B_{11}} = 609.9$$

$$\beta = \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} = 614.3$$

$$\gamma = -B_{12}\alpha^2\beta/\lambda = 15.2$$

$$u_0 = \gamma v_0 / \beta - B_{13}\alpha^2/\lambda = 344.7$$

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 609.9 & 15.2 & 344.7 \\ 0 & 614.3 & 219.2 \\ 0 & 0 & 1 \end{bmatrix}$$

2.2 d) "Explain each entry in A. Perfect camera, why not perfect?"

u_0, v_0 : coordinate for principal point

α, β : scale factors in image u, v axes

γ : describing skewness of two image axes.

A perfect camera would have same values for focal lenght and 0 for γ.

A non zero γ could suggest misalinment with camera's imaging system.

Other reasons: calibration error, lens imperfation, sensor characteristics.

```
clc; clear;

% Load the needed data from problem 2.1
load('image_coordinates.mat');  % Load u and v
load('world_coordinates.mat');  % Load X and Y
load('homography_matrix.mat');  % Load H
load('constants.mat');          % Load constants

% Stacking the Li to get the L
L = [];
H_original = H;
for img = 1:numImages
    % Extract the H matrix for the given image
    H_vector = H_original(:,img);
    H_I = reshape(H_vector,3,3).';

    % Construct L_i
    L_i =[H_I(1,1)*H_I(1,2) H_I(2,1)*H_I(1,2)+H_I(1,1)*H_I(2,2) H_I(3,1)*H_I(1,2)↙
+H_I(1,1)*H_I(3,2) H_I(2,1)*H_I(2,2) H_I(3,1)*H_I(2,2)+H_I(2,1)*H_I(3,2) H_I(3,1)↙
*H_I(3,2);
         H_I(1,1)^2-H_I(1,2)^2 2*H_I(2,1)*H_I(1,1)-2*H_I(2,2)*H_I(1,2) 2*H_I(3,1)↙
*H_I(1,1)-2*H_I(3,2)*H_I(1,2) H_I(2,1)^2-H_I(2,2)^2 2*H_I(3,1)*H_I(2,1)-2*H_I(3,2)↙
*H_I(2,2) H_I(3,1)^2-H_I(3,2)^2];
    L = [L; L_i];
end

% Solve the equation L * b = 0
[~, ~, V] = svd(L);
b = V(:, end);

% Solving the equations from Appendix B
b11 = b(1); b12 = b(2); b13 = b(3);
b22 = b(4); b23 = b(5); b33 = b(6);

v_0 = (b12*b13-b11*b23)/(b11*b22-b12^2);
lambda = b33-(b13^2 + v_0*(b12*b13-b11*b23))/b11;
alpha = sqrt(lambda/b11);
beta = sqrt(lambda*b11/(b11*b22-b12^2));
gamma = -b12*alpha^2*beta/lambda;
u_0 = gamma*v_0/beta - b13*alpha^2/lambda;

A = [alpha gamma u_0;
      0    beta  v_0;
      0      0     1];
save('Extrinsic_matrix.mat', 'A');
```

3 "Calculating Extrinsic Matrix from Intrinsic Matrix"

a) "Given H and A calculate extrinsic matrix [R|t] for each image"

From section 3.1 we have:

$$[R|t] = [\underline{r_1}\ \underline{r_2}\ \underline{r_3}\ |\underline{t}]$$

$$\underline{r_1} = \lambda A^{-1} \underline{h_1}$$

$$\underline{r_2} = \lambda A^{-1} \underline{h_2} \qquad \lambda = \frac{1}{\|A^{-1} \underline{h_1}\|}$$

$$\underline{r_3} = \underline{r_1} \times \underline{r_2}$$

$$\underline{t} = \lambda A^{-1} \underline{h_3}$$

I solve this in Matlab same way as before. Image for image. See pdf of code.

(I do the same for the other 8).

For Image 1 I get:

$$[R|t] = \left[\begin{array}{ccc|c} 0.992 & 0.112 & 0.033 & -170 \\ -0.093 & 0.567 & 0.818 & -34 \\ 0.077 & -0.815 & 0.573 & 514 \end{array}\right]$$

3 b) "Find trajectory"

$$RC + t = 0 \quad | \quad C: \text{camera position.}$$

Solve for C:

$$C = -R^{-1}t$$

We have R and t for each photo and I solve in Matlab and get camera positions:

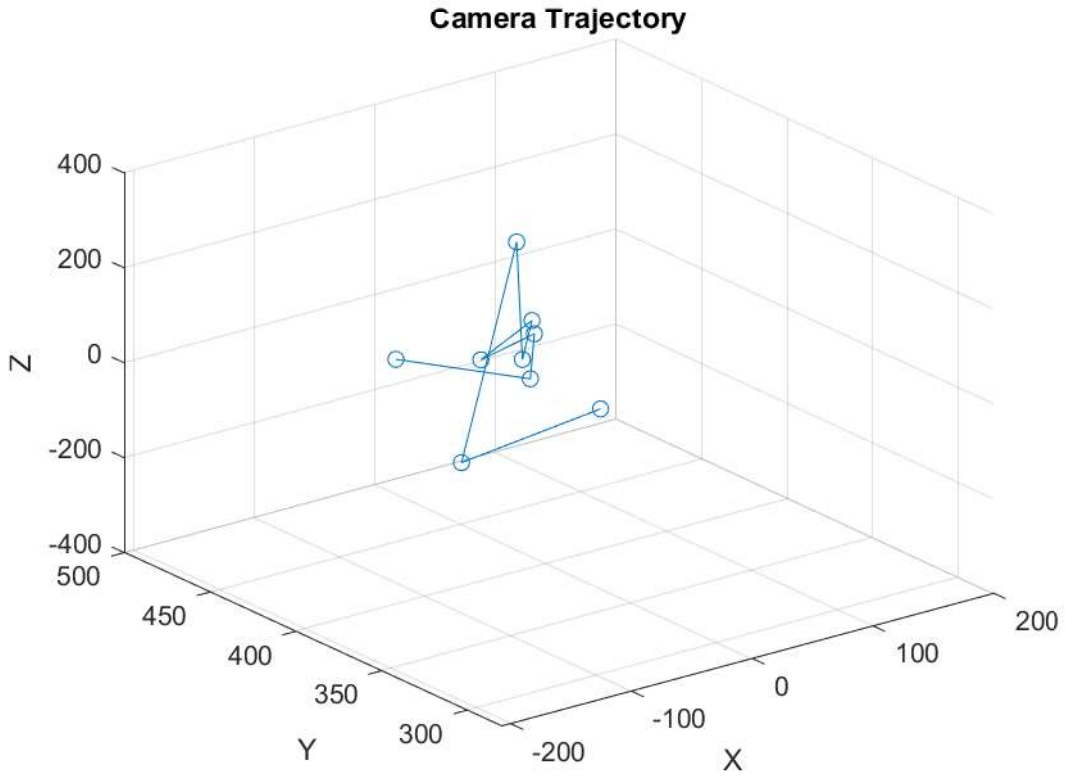
① $\begin{bmatrix} 129 \\ 459 \\ -261 \end{bmatrix}$ ② $\begin{bmatrix} 1 \\ 450 \\ -272 \end{bmatrix}$ ③ $\begin{bmatrix} -41 \\ 388 \\ 325 \end{bmatrix}$

④ $\begin{bmatrix} -140 \\ 315 \\ 267 \end{bmatrix}$ ⑤ $\begin{bmatrix} -127 \\ 319 \\ 335 \end{bmatrix}$ ⑥ $\begin{bmatrix} -180 \\ 311 \\ 300 \end{bmatrix}$

⑦ $\begin{bmatrix} -141 \\ 308 \\ 334 \end{bmatrix}$ ⑧ $\begin{bmatrix} -184 \\ 279 \\ 316 \end{bmatrix}$ ⑨ $\begin{bmatrix} -208 \\ 341 \\ 271 \end{bmatrix}$

See plots in pdf

```
clear; clc;

% Load the needed data from problem 2.1
load('image_coordinates.mat');  % Load u and v
load('world_coordinates.mat');  % Load X and Y
load('homography_matrix.mat');  % Load H
load('constants.mat');          % Load constants
load('Extrinsic_matrix.mat');   % Load Extrinsic

% Initialize variables to store camera positions
camera_positions = zeros(3, numImages);

R_all = [];
% Loop through all images to calculate [R|t] for each image
for img = 1:numImages
    % Extract the H matrix for the given image
    H_vector = H(:, img);
    h1 = [H_vector(1); H_vector(4); H_vector(7)];
    h2 = [H_vector(2); H_vector(5); H_vector(8)];
    h3 = [H_vector(3); H_vector(6); H_vector(9)];

    % Compute the lambda
    lambda = 1 / norm(A^-1 * h1);

    % Compute r1, r2, r3, t
    r1 = lambda * (A^-1) * h1;
    r2 = lambda * (A^-1) * h2;
    r3 = cross(r1, r2);
    t = lambda * (A^-1) * h3;

    % Combine R and t to create [R|t]
    R = [r1 r2 r3];
    R_all = [R_all; R];

    % Store the camera position (C) in the camera_positions array
    camera_positions(:, img) = - R^(-1) * t;

end

% Plot the camera trajectory
figure;
plot3(camera_positions(1,:),camera_positions(2,:),camera_positions(3,:),'-o');
grid on;
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Camera Trajectory');
```


Camera Trajectory
400
200
0
-200
-400
Z
500
450
400
350
300
Y
-200
-100
0
100
200
X