CS-E4850 Computer Vision

Exercise Round 12

Laura Alejandra Encinar Gonzalez

101583950

Exercise 2

Exercise 2.

$\overline{O'p'}^T \cdot (\overline{OO'} \times \overline{Op}) = 0 \Leftrightarrow \overline{O'p'}, \overline{OO'}$ and $\overline{Op}$ are coplanar

and $\overline{OO'} = t = (t_1, t_2, t_3)$ translation between cameras

So $\overline{O'p'}$, t and $(\overline{O'p'} - t)$ are also coplanar

$\Leftrightarrow \overline{O'p'}^T \cdot (t \times (\overline{O'p'} - t)) = 0$

Since the camera calibration matrix K is the identity matrix $K = I$, the homogeneous coordinate vectors x and x' represent the incoming light rays in the camera coordinate frame.

$$x'^T \cdot (t \times (x' - t)) = 0$$

If we write x' like a transformation of vector x using $P' = [R\ t]$, we get : $x' = Rx + t$

$$x'^T (t \times (Rx + t - t)) = 0$$

$$x'^T (t \times (Rx)) = 0$$

$$x'^T ([t]_\times R) \cdot x = 0$$

$$\boxed{x'^T E x = 0}$$

Exercise 3

Exercise 3

a) $$\text{disparity} = x - x' = \frac{b \cdot f}{Z_P} \iff Z_P = \frac{b \cdot f}{d} = \frac{6 \cdot 1}{1} = 6 \text{ cm}$$

$$\boxed{Z_P = 600 \text{ mm}}$$

b) disparity < 1 pixel = 0.01 mm

$$\frac{b \cdot f}{Z_P} < 0.01 \iff 0.01 \cdot Z_P > 600 \cdot 100 \iff \boxed{Z_P > 6\,000\,000 \text{ mm}}$$

c)

- Image of Q on the image plane of the left camera:

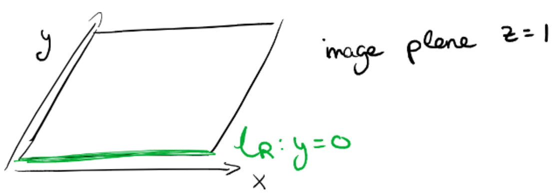
$$x_l = P_l \cdot Q = [I \;\; 0] \cdot Q = \boxed{(3 \;\; 0 \;\; 1)}$$

- Epipolar line on the image plane of the right camera:

$$l_R = E \cdot x_l = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \cdot \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 6 \\ 0 & -6 & 0 \end{bmatrix} \cdot \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 6 \\ 0 \end{pmatrix}$$

$$l_R : 6y = 0$$

$$\boxed{l_R : y = 0}$$



Exercise 4

a) Implement the eight-point algorithm

```
function F = estimateF(pts1, pts2)
    % Implement the eight-point algorithm
    A = zeros(size(pts1, 1), 9);

    for i = 1:size(pts1, 1)
        A(i, :) = kron(pts1(i, :), pts2(i, :));
    end

    [~, ~, V] = svd(A);
    F = reshape(V(:, end), 3, 3)';

    % Ensure rank 2
    [U, S, V] = svd(F);
    S(3, 3) = 0;
    F = U * S * V';

end
```

b) Implement the normalized eight-point algorithm

```
function Fnorm = estimateFnorm(pts1, pts2)

    % Normalize coordinates
    [pts1, T1] = normalizePoints(pts1);
    [pts2, T2] = normalizePoints(pts2);

    Fnorm = estimateF(pts1, pts2);

    % Denormalize Fnorm
    Fnorm = T2' * Fnorm * T1;
end
```

```
function [ptsNormalized, T] = normalizePoints(pts)
    % Normalize points by scaling and translating to have zero mean and
    % average distance 2 from the origin

    % Compute mean of points
    meanPts = mean(pts);

    % Translate points to have zero mean
    ptsCentered = pts - meanPts;

    % Compute average distance from the origin
    avgDist = 2/mean(sqrt(ptsCentered(:,1).^2 + ptsCentered(:,2).^2));

    % Scale points
    ptsNormalized = ptsCentered * avgDist;
    ptsNormalized(:,3)=1;

    % Transformation matrix for denormalization
    T = [avgDist, 0, -meanPts(1)*avgDist;
         0, avgDist, -meanPts(2)*avgDist;
         0, 0, 1];

end
```

Results: