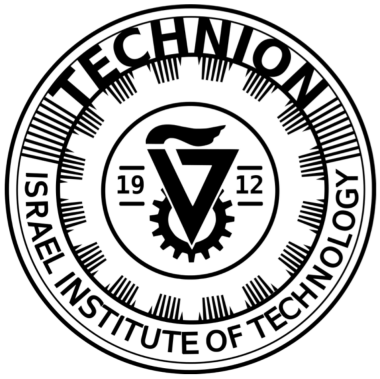
Technion - Israel Institute of Technology

$\mathbf{Hw_1}$ - Photometry, Geometry and Camera calibration

---

Chen Katz, ID 203511043

Daniel Engelsman, ID 300546173



שב1 - שאלה 1

שבת 30 נובמבר 2019 16:15

<u>נתון:</u>

$R = 10$ הכדור למברטי

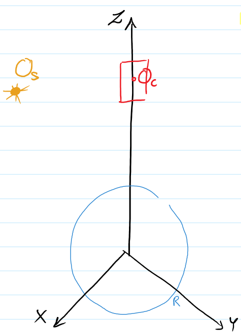
$O_b = O_{ball} = (0,0,0)^{world}$

$O_c = O_{camera} = (0,0,100)^{world}$

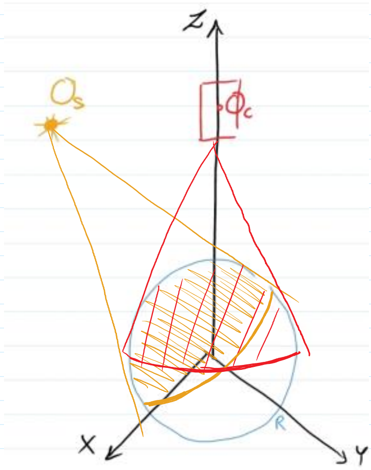
מצלמה אידיאלית מסתכלת לכיוון z השלילי

$O_s = O_{source} = (L_x, L_y, L_z > 0)^{world}$

A.



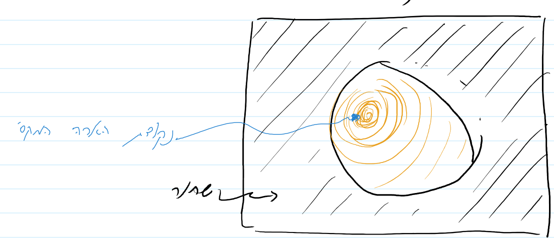
B.



האזור שמסומן בכתום (▨) הוא האזור שאותו מאיר מקור האור.
האזור שמסומן באדום (▨) הוא האזור בכדור שאותו המצלמה יכולה לקלוט.

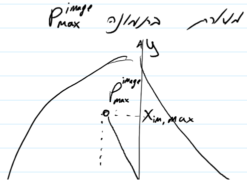
⇐
האזור שמסומן בכתום ואדום יראה בתמונה
האזור שמסומן באדום בלי כתום לא יראה בתמונה (כי מקור האור הוא יחיד)

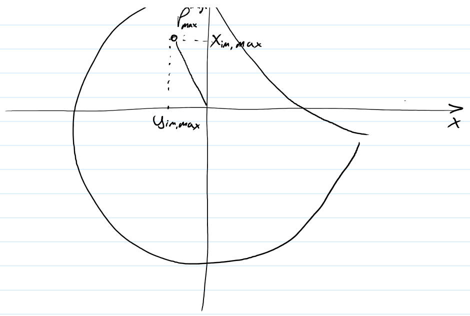
לכן התמונה שתתקבל תראה ככה:



C. <u>אלגוריתם למציאת כיוון הארה:</u>

- מציאת הנקודה הכי מוארת בתמונה P^{image}_{max}





הכדור למברטי ולכן בנקודה זו הנורמל לכדור מקביל לכיוון למקור האור

וזה כי $I_{max}=\rho\cdot n\cdot s$, לכן מציאת כיוון הנורמל שקולה למציאת הכיוון למקור האור.

- המצלמה אידיאלית ולכן:

$$\begin{cases} x_{im} = \frac{X^c}{Z^c} = \frac{X^w}{Z^w+100} \\ y_{im} = \frac{Y^c}{Z^c} = \frac{Y^w}{Z^w+100} \end{cases}$$

רדיוס הכדור ידוע:

$$(X^w)^2+(Y^w)^2+(Z^w)^2=10^2$$

3 משוואות ב 3 נעלמים - נמצא את P_{max}^{world} התלת מימדית.

- כיוון מקור האור בקואורדינטות של העולם הוא: $\frac{P_{max}^{world}}{\|P_{max}^{world}\|}$

שבו1 - שאלה 2

שבת 23 נובמבר 2019   19:29

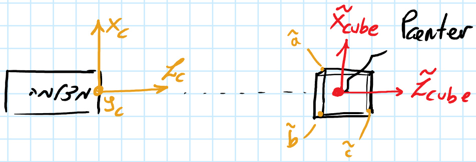
נתון:

$$\begin{cases} F = 1[m] \\ c_x = c_y = 0 \end{cases} \Rightarrow P_{sensor} = \begin{pmatrix} \frac{X_{camera}}{Z_{camera}} \\ \frac{Y_{camera}}{Z_{camera}} \end{pmatrix}$$

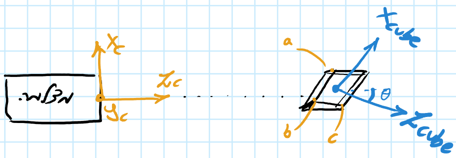
$$P_{center} = (0, 0.5, 10)\ [m]$$

גודל הקוביה $1[m^3]$ ⇐ אורך צלע הקוביה $1[m]$

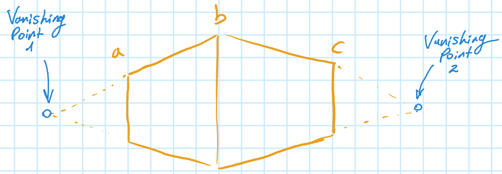
ציור של הקוביה לפני הסיבוב:



ציור של הקוביה לאחר הסיבוב:



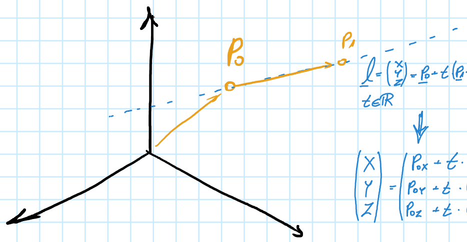
במישור התמונה הקוביה תראה ככה:



דרך למצוא את נקודות הבריחה?

$$\begin{cases} X_{image} = 1 \cdot \frac{X_{cube}}{Z_{cube}} \\ Y_{image} = 1 \cdot \frac{Y_{cube}}{Z_{cube}} \end{cases}$$

ניתן להציג ישר פרמטרי בדרך הבאה ע"י 2 נקודות (שידועים נקודה ווקטור כיוון):



$$l = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = P_0 + t(P_1 - P_0) = \begin{pmatrix} P_{0x} \\ P_{0y} \\ P_{0z} \end{pmatrix} + t \cdot \begin{pmatrix} P_{1x} - P_{0x} \\ P_{1y} - P_{0y} \\ P_{1z} - P_{0z} \end{pmatrix}, \quad t \in \mathbb{R}$$

$$\Downarrow$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} P_{0x} + t \cdot (P_{1x} - P_{0x}) \\ P_{0y} + t \cdot (P_{1y} - P_{0y}) \\ P_{0z} + t \cdot (P_{1z} - P_{0z}) \end{pmatrix}$$

הישר הפרמטרי במישור התמונה:

$$X_{image} = \frac{X_{cube}}{Z_{cube}} = \frac{P_{0x} + t(P_{1x} - P_{0x})}{P_{0z} + t(P_{1z} - P_{0z})}$$

☐ מכיוון שלכל הקווים המקבילים יש את אותו

$$X_{image} = \frac{X_{cube}}{Z_{cube}} = \frac{P_{0x} + t(P_{1x} - P_{0x})}{P_{0z} + t(P_{1z} - P_{0z})}$$

$$Y_{image} = \frac{Y_{cube}}{Z_{cube}} = \frac{P_{0Y} + t(P_{1Y} - P_{0Y})}{P_{0z} + t(P_{1z} - P_{0z})}$$

מכיוון שלכל הקווים המקבילים יש את אותו וקטור כיוון $v = \begin{pmatrix} P_{1x} - P_{0x} \\ P_{1y} - P_{0y} \\ P_{1z} - P_{0z} \end{pmatrix}$ נקודת ההיעלמות של כל הקווים המקבילים זהה.

נקודת ההיעלמות היא הנקודה המרוחקת ביותר במישור התמונה והיא מתקבלת עבור $t \to \infty$:

$$\begin{cases} X_{vp} = \lim_{t\to\infty} \left(\frac{P_{0x} + t(P_{1x} - P_{0x})}{P_{0z} + t(P_{1z} - P_{0z})} \right) = \frac{P_{1x} - P_{0x}}{P_{1z} - P_{0z}} \\ Y_{vp} = \lim_{t\to\infty} \left(\frac{P_{0Y} + t(P_{1Y} - P_{0Y})}{P_{0z} + t(P_{1z} - P_{0z})} \right) = \frac{P_{1Y} - P_{0Y}}{P_{1z} - P_{0z}} \end{cases}$$

$$\text{Vanishing Point} = \frac{1}{P_{1z} - P_{0z}} \cdot (P_{1x} - P_{0x}, P_{1Y} - P_{0Y})$$

מציאת הנקודות במישור הקוביה:

מכיוון שידוע אורך צלע הקוביה ומרכז הקוביה לפני הסיבוב ניתן לחשב את הנקודות במישור cube (שמצויר בורוד למעלה) ואז בעזרת מטריצת סיבוב להעביר למישור cube מצויר בכחול למעלה)

טרנספורמציה לנקודה: $P_{cube} = R_Y(\theta) \cdot P_{\tilde{cube}} + t$

הסיבוב הוא סביב ציר Y בזווית θ-

טרנספורמציה לוקטור: $V_{cube} = R_Y(-\theta) \cdot V_{\tilde{cube}}$ כי מה שמעניין אותנו.

$$R_Y(-\theta) = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix}$$

צריך למצוא 2 נקודות היעלמות לכן צריך 2 וקטורים

$$\overline{\tilde{b}\tilde{a}} = (1, 0, 0)$$

$$\overline{\tilde{b}\tilde{c}} = (0, 0, 1)$$

$$\overline{ba} = R_Y(-\theta) \cdot \overline{\tilde{b}\tilde{a}} = \begin{pmatrix} \cos(\theta) \\ 0 \\ \sin(\theta) \end{pmatrix}$$

$$\overline{bc} = R_Y(-\theta) \cdot \overline{\tilde{b}\tilde{c}} = \begin{pmatrix} -\sin(\theta) \\ 0 \\ \cos(\theta) \end{pmatrix}$$

חישוב נקודות ההיעלמות:

$$VP1 = \left(\frac{\cos(\theta)}{\sin(\theta)}, \frac{0}{\sin(\theta)} \right) = \left(\frac{\cos(\theta)}{\sin(\theta)}, 0 \right)$$

$$VP2 = \left(-\frac{\sin(\theta)}{\cos(\theta)}, \frac{0}{\cos(\theta)} \right) = \left(-\frac{\sin(\theta)}{\cos(\theta)}, 0 \right)$$

שב 1 - שאלה 3

יום ראשון 24 נובמבר 2019 20:29

(A) נדרש להראות שקווים ישרים (תלת מימדיים) בעולם מועלים לקווים ישרים בתמונה.

__פתרון:__

ייצוג של קו ישר 3D:

$$\frac{x-x_0}{a}=\frac{y-y_0}{b}=\frac{z-z_0}{c}$$

$$\Downarrow$$

$$\begin{cases} x-x_0=\frac{a}{c}(z-z_0) \\ y-y_0=\frac{b}{c}(z-z_0) \end{cases}$$

$$\Downarrow$$

$$\begin{cases} x=\underbrace{\tfrac{a}{c}}_{m_x}z+\underbrace{(x_0-\tfrac{a}{c}z_0)}_{b_x} \\ y=\underbrace{\tfrac{b}{c}}_{m_y}z+\underbrace{(y_0-\tfrac{b}{c}z_0)}_{b_y} \end{cases}$$

$$\Downarrow$$

$$(*)\quad \begin{cases} x=m_x\cdot z+b_x \\ y=m_y\cdot z+b_y \end{cases}$$

נקודה בקואורדינטות המצלמה כפי שנראים בתמונה:

$$(**)\quad \begin{cases} X_{image}=f_x\cdot\frac{x}{z}+c_x \\ y_{image}=f_y\cdot\frac{y}{z}+c_y \end{cases}$$

נחבר את $(*)$ ו $(**)$:

$$\begin{cases} X_{image}=f_x\cdot\frac{m_x\cdot z+b_x}{z}+c_x=f_x m_x+\frac{f_x}{z}\cdot b_x+c_x \\ y_{image}=f_y\cdot\frac{m_y\cdot z+b_y}{z}+c_y=f_y\cdot m_y+\frac{f_y}{z}\cdot b_y+c_y \end{cases}$$

$$\Downarrow$$

$$(*)\quad \begin{cases} X_{image}-f_x m_x-c_x=\frac{f_x}{z}\cdot b_x \\ y_{image}-f_y m_y-c_y=\frac{f_y}{z}\cdot b_y \end{cases}$$

$$\Downarrow$$

$$\frac{X_{image}-f\cdot m_x-c_x}{y_{image}-f\cdot m_y-c_y}=\frac{b_x}{b_y}$$

$$\Downarrow$$

$$X_{image}=\underbrace{\frac{b_x}{b_y}}_{m}\cdot y_{image}+\underbrace{\left[\frac{b_x}{b_y}(-f\cdot m_y-c_y)-f\cdot m_x+c_x\right]}_{b}$$

$$\Downarrow$$

$$X_{image}=m\cdot y_{image}+b$$

קו ישר במישור התמונה

__מקרה מיוחד:__ כאשר $b_x=b_y=0$ – הישר __עובר__ דרך $(0,0,0)$ $\begin{pmatrix} x=m_x\cdot z \\ y=m_y\cdot z \\ z=z \end{pmatrix}$ מתקבל מ $(*)$ ש $\begin{cases} X_{im}=f_x\cdot m_x+c_x \\ y_{im}=f_y\cdot m_y+c_y \end{cases}$ - כל הנקודות על הישר התלת מימדי מתקבלות בנקודה בתמונה

(B) נדרש להראות שקווים מקבילים (תלת מימדיים) בעולם מועלים לתמונה ומתכנסים בנקודת ההיעלמות

__פתרון:__

משפחה של קווים ישרים מקבילים 3D:

$$\left\{\bar{P}=\bar{P}_0+t\cdot\bar{v}\;\middle|\; \begin{matrix} \bar{v}\in\mathbb{R}^3 \text{ קבוע} \\ \bar{P}_0\in\mathbb{R}^3 \text{ שונה עבור כל ישר במשפחה} \\ \forall t\in\mathbb{R} \end{matrix}\right\}=\left\{\underbrace{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}}_{\bar{P}}=\underbrace{\begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix}}_{\bar{P}_0}+t\underbrace{\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}}_{\bar{V}}\;\middle|\; \begin{matrix} V_x,V_y,V_z\in\mathbb{R} \text{ קבועים} \\ X_0,Y_0,Z_0\in\mathbb{R} \text{ שונים עבור כל ישר במשפחה} \\ \forall t\in\mathbb{R} \end{matrix}\right\}$$

נקודה בקואורדינטות המצלמה כפי שנראים בתמונה:

$$\begin{cases} X_{image}=f_x\cdot\frac{X}{Z}+c_x \\ y_{image}=f_y\cdot\frac{Y}{Z}+c_y \end{cases}$$

עבור $\bar{v}$ כלשהו, ישר מהמשפחה של הישרים המקבילים כפי שנראה בתמונה:

$$\left\{\begin{array}{l} X_{image}=f_x\cdot\frac{X_0+t\cdot V_x}{Z_0+t\cdot V_z}+c_x \\ Y_{image}=f_y\cdot\frac{Y_0+t\cdot V_y}{Z_0+t\cdot V_z}+c_y \end{array}\ \middle|\ \forall t\in\mathbb{R}\right\}$$

עבור $t\to\infty$ כל הישרים המקבילים מתכנסים בנקודת ההעלמות:

$$\begin{cases} X_{vp}=\lim_{t\to\infty}\left(f_x\cdot\frac{X_0+t\cdot V_x}{Z_0+t\cdot V_z}+c_x\right)=f_x\cdot\frac{V_x}{V_z}+c_x \\ Y_{vp}=\lim_{t\to\infty}\left(f_y\cdot\frac{Y_0+t\cdot V_y}{Z_0+t\cdot V_z}+c_y\right)=f_y\cdot\frac{V_y}{V_z}+c_y \end{cases}$$

$$\Downarrow$$

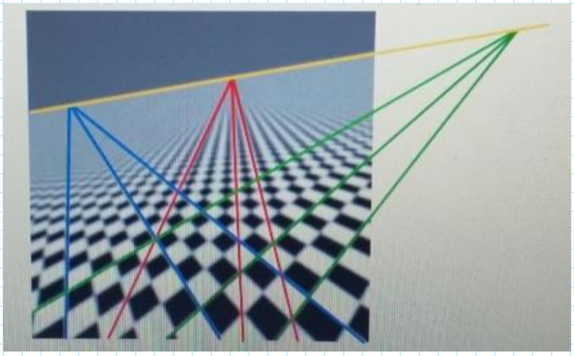
$$\bar{P}_{vp}=\left(f_x\cdot\frac{V_x}{V_z}+c_x,\ f_y\cdot\frac{V_y}{V_z}+c_y\right)$$

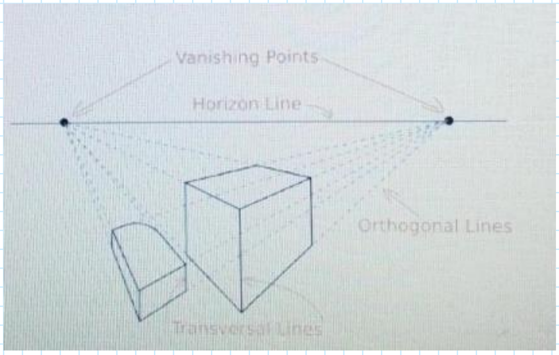
מקרה מיוחד: כאשר $V_z=0$ (כאשר הקו מקביל למישור XY - מישור התמונה) - במקרה הזה נקודת ההעלמות לא מוגדרת ולא תתקבל.

(c) נדרש להראות שכל נקודות ההעלמות של קווים מקבילים שהם מקבילים למישור מונחות על אותו קו ישר

פתרון:

ציור להדגמה:





ניקח משפחת הקווים שבאותו מישור:

בסעיף הקודם הראנו שכל קו ממשפחת הקווים 3D באותו כיוון $\bar{v}$ מסוים מתכנסים לאותה נקודת העלמות במישור התמונה (כי קיבלנו שנקודת ההעלמות לא תלויה בנקודה P_0 על הישר אלא רק בווקטור הכיוון $\bar{v}$).

משפחת קווים ישרים בכיוון $\bar{v}_j$ מסוים:

$$M_j=\left\{P_0+\bar{v}^j\cdot t\ \middle|\ \begin{array}{l}\forall P_0\in\mathbb{R}^3\\ t\in\mathbb{R}\\ \bar{v}^j\in\mathbb{R}^3\end{array}\right\}$$

נסמן ב M_j את כל משפחות הקווים הישרים בכיוון $\bar{v}^j$ שבהם $\bar{v}^j$ משיקים למישור.

מכיוון ש v_j משיק למישור הוא מאונך לנורמל שלו: $\hat{n}\cdot\bar{v}^j=0 \Leftrightarrow n_x\cdot V_x+n_y\cdot V_y+n_z\cdot V_z=0$

$$n_x\cdot V_x^j=-\left(n_y\cdot V_y^j+n_z\cdot V_z^j\right)$$

$$V_x^j=-\frac{1}{n_x}\left(n_y\cdot V_y^j+n_z\cdot V_z^j\right)$$

$$\frac{V_x^j}{V_z^j}=-\frac{1}{n_x}\left(\frac{n_y\cdot V_y^j}{V_z^j}+n_z\right)=-\frac{n_y}{n_x}\cdot\frac{V_y^j}{V_z^j}+n_z$$

לפי הסעיף הקודם - קו ממשפחת הקווים M_j יתכנס בנקודת ההעלמות הבאה:

$$\bar{P}_{vp-j}=\left(f_x\cdot\frac{V_x^j}{V_z^j}+c_x,\ f_y\cdot\frac{V_y^j}{V_z^j}+c_y\right)=\Big(\underbrace{-f_x\frac{n_y}{n_x}\cdot\frac{V_y^j}{V_z^j}+f_x\cdot n_z+c_x}_{=X_{vp-j}},\ \underbrace{f_y\cdot\frac{V_y^j}{V_z^j}+c_y}_{=Y_{vp-j}}\Big)$$

$= x_{vp\text{-}j}$ $\quad$ $-y_{vp\text{-}j}$

מיקום נקודת העלמות של משפחת הישרים M_j תלויה בפרמטר $\alpha^j = \frac{V_y^j}{V_z^j}$.

נראה שכל נקודות ההעלמות מתקבלות על אותו ישר במישור התמונה ($x_{vp}^j = m \cdot y_{vp}^j + n$)

$$x_{vp}^j = -f_x \cdot \frac{n_y}{n_x} \cdot \alpha^j + f_x \cdot n_z + c_x$$

$$y_{vp}^j = f_y \cdot \alpha^j + c_y \rightarrow \alpha_j = \frac{y_{vp}^j - c_y}{f_y}$$

$$x_{vp}^j = -\frac{f_x}{f_y} \cdot \frac{n_y}{n_x} \cdot (y_{vp}^j - c_y) + f_x \cdot n_z + c_x$$

$$x_{vp}^j = \underbrace{-\frac{f_x}{f_y} \cdot \frac{n_y}{n_x}}_{m} \cdot y_{vp}^j + \underbrace{\frac{f_x}{f_y} \cdot \frac{n_y}{n_x} \cdot c_y + f_x \cdot n_z + c_x}_{n}$$

$$x_{vp}^j = m \cdot y_{vp}^j + n$$

$\Downarrow$

כל ישרים שמקבילים למישור שהגדרנו שלו $\hat{n}$ יתקבלו על אותו ישר

<u>מקרה חריג</u>: כמו בסעיפים הקודמים - כאשר $V_z=0$ לא מתקבלת נקודת העלמות.

1 Question

a. Compute the camera matrix M which generated the image points x according to the camera equation.

   Use the linear DLT method which was presented in the tutorial.

b. Re-project all the real world point X to their estimated corresponded points $\tilde{x}$ on the image plane using the estimated matrix M. Define a way to compute the estimation error based on $\tilde{x}$. Explain the error measure you defined. What are its units?

We will now determine the intrinsic parameters K and the extrinsic parameters R and X_0

c. Use the given function "rq.m" to reconstruct R, K .

a. The projection matrix M satisfies the relation - $\tilde{x}_{\text{image}} = \mathbf{M} \cdot \mathbf{X}_{3D}$. Having not knowing its values, one has to build it relying on the a-priori knowledge of the 3D world points and their corresponding pixels on the image :

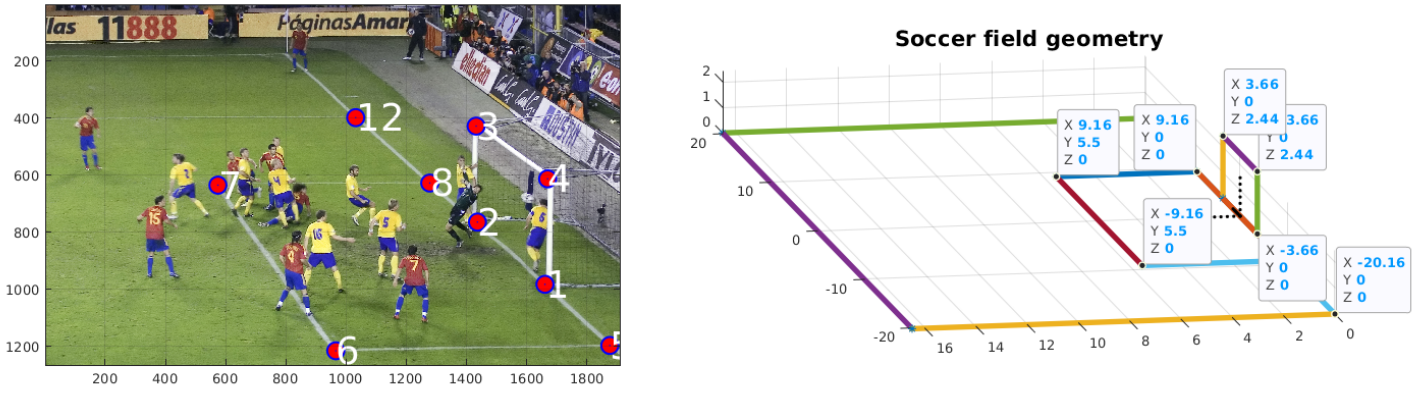


Figure 1: Matching corresponding $\{\tilde{x}_i, X_i\}$

Now, using the DLT method (see code : $DLT.m$) :

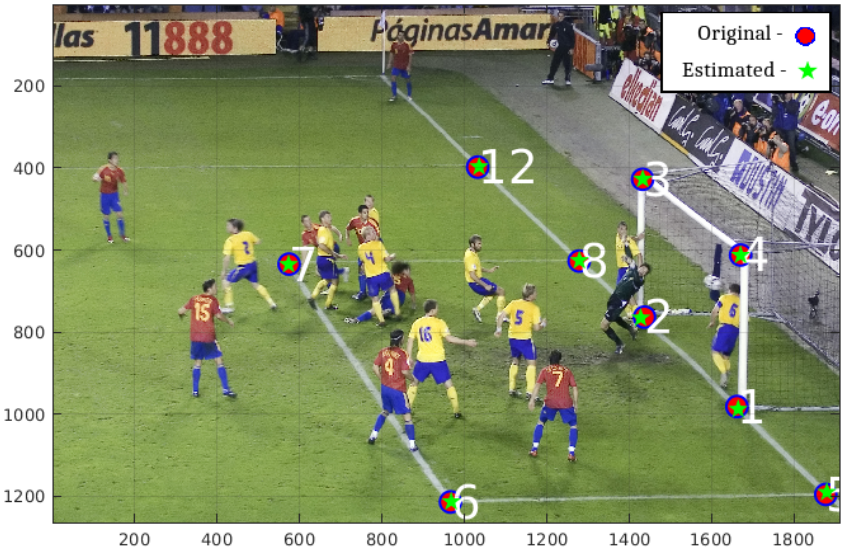
```
30 -    x = Pts2d( :, all( ~isnan(Pts2d) ) );          % Extract valid (in-frame) points
31 -    n_dim = length(x);                             % Optional :: reduce number of points
32 -    m = DLT(X, x, n_dim);                          % Singular Values of M
33 -    M = reshape(m, 4, 3)';                         % Reshape into desired dimensions
```

```
Command Window
New to MATLAB? See resources for Getting Started.

M =

    0.0057    0.0823   -0.0010   -0.8644
    0.0100   -0.0001    0.0820   -0.4891
   -0.0000    0.0000    0.0000   -0.0006
```

b. Having M computed, we'll use it for projecting the 3D points onto the image plane :



The slight differences between the pixels centres (◯ vs. ⋆) can be explained by the numerical accuracy of the algorithm, and the number of equations that are used (here 9). Given 11 unknown variables in M matrix, the minimal number of equations should be 6, but the more correspondences are used, the more accurate M will be.

The estimation error can be calculated in few ways ($\bar{\epsilon} = [\Delta u, \Delta v]$) :

i. Mean Absolute Error - $\frac{1}{n}\sum_{j=1}^{n}|x_i - x_{est}| = [2.4184,\ 2.0808]$ [px]

ii. Root Mean Square Error - $\sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_i - x_{est})^2} = [3.5947,\ 2.9566]$ [px]

iii. Relative Error - $\left|\frac{\bar{x} - \bar{x}_{est}}{\bar{x}}\right| = \left|1 - \frac{\bar{x}_{est}}{\bar{x}}\right| = [0.0075,\ 0.0178]$ [%]

c. Use the given function "rq.m" to reconstruct R , K :

K

3x3 double

	1	2	3
1	0.0819	5.7587e-04	0.0100
2	0	0.0822	-0.0086
3	0	0	8.1900e-06

R

3x3 double

	1	2	3
1	0.1897	0.9816	-0.0216
2	0.0189	0.0183	0.9997
3	-0.9817	0.1900	0.0151

1.1 Describe the goal of this function :
Performing QR factorization of $M = QT$ where (Q $\rightarrow$ orthogonal, T $\rightarrow$ upper triangular).

1.2 What is the difference between the given function and the matlab function called "qr.m"? Why didn't we use the qr function?

qr.m (**2.i**) performs decomposition which is opposited to what we need (elaborated next). It is therefore why we should use *rq.m* algorithm to set it right (see **2.ii**).

2. Explain the operation done at each of the function's lines :
i. qr.m (*Wikipedia*) : Let $\mathbf{M} = [m_1,\ ...,\ m_n]$ be a full column rank matrix, with inner product $\langle \mathbf{v}, \mathbf{w} \rangle = \mathbf{v}^T \mathbf{w}$. Define the projection :

$$\text{proj}_u m = \frac{\langle \mathbf{u}, \mathbf{m} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u} \quad \text{such that,}$$

$$\mathbf{u}_k = \mathbf{m}_k - \sum_{j=1}^{k-1} \text{proj}_{\mathbf{u}_j}\ \mathbf{m}_k, \quad \mathbf{e}_k = \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|} \quad \forall \quad k$$

$\mathbf{m}_i$'s can now be expressed over our newly computed orthonormal basis :

$$\mathbf{m}_k = \sum_{j=1}^{k} \langle \mathbf{e}_j, \mathbf{m}_k \rangle \mathbf{e}_j \quad \text{where} \quad \langle \mathbf{e}_i, \mathbf{a}_i \rangle = \|\mathbf{u}_i\| \quad \text{such that - } A = QR$$

$$\text{where} \quad Q = [e_1,\ ...,\ e_n], \quad \text{and - } R = \begin{pmatrix} \langle \mathbf{e}_1, \mathbf{a}_1 \rangle & \langle \mathbf{e}_1, \mathbf{a}_2 \rangle & \langle \mathbf{e}_1, \mathbf{a}_3 \rangle & \dots \\ 0 & \langle \mathbf{e}_2, \mathbf{a}_2 \rangle & \langle \mathbf{e}_2, \mathbf{a}_3 \rangle & \dots \\ 0 & 0 & \langle \mathbf{e}_3, \mathbf{a}_3 \rangle & \dots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

ii. rq.m using the above **qr.m** algorithm on M, we will then handle Q, R as :

$$T_{QR}\ : M^T = \begin{cases} R = \text{flip}_{lr}\ (\ \text{flip}_{ud}\ (R^T)\ ) \\ Q = \text{flip}_{ud}\ (\ Q^T\ ) \end{cases} \quad \Rightarrow \quad \begin{cases} R = R \cdot \text{diag}(\ \text{sgn}(\ R\ )\ ) \\ Q = T \cdot Q \end{cases}$$

d. What can you say about the camera's characteristics, from the matrix K ? As seen at tutorials, the intrinsic matrix should be eventually as such :

$$K = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix}$$

Therefore we should normalize our K matrix so we will get :

$$K = \begin{pmatrix} 0.0819 & 5.7557\text{e-}4 & 0.01 \\ 0 & 0.0822 & -0.0086 \\ 0 & 0 & 8.19\text{e-}6 \end{pmatrix} \cdot \frac{1}{K(3,3)} \Rightarrow K = \begin{pmatrix} 1.0005e4 & 70.3140 & 1.2237\text{e+}3 \\ 0 & 1.0032\text{e+}4 & -1.0531\text{e+}3 \\ 0 & 0 & 1 \end{pmatrix}$$

e. What can you say about the orientation of the camera from the matrix R ?

Our rotation matrix satisfies $det(R) = -1$, which is incompatible with the currnet right handed coordinate system - $det(R) = 1$, so we'll multiply by -1 and get :

$$R = \begin{pmatrix} -0.1897 & -0.9816 & 0.0216 \\ -0.0189 & -0.0183 & -0.9997 \\ 0.9817 & -0.1900 & -0.0151 \end{pmatrix}$$

(!) Using *rot2eul.m* from matlab, we get the following body (Euler) angles :

$$[\phi,\ \theta,\ \psi] \ = \ [-89.1344 - 1.2375 - 79.0648]$$

Which follow 3-axis consecutive 'x-y-z' rotations.

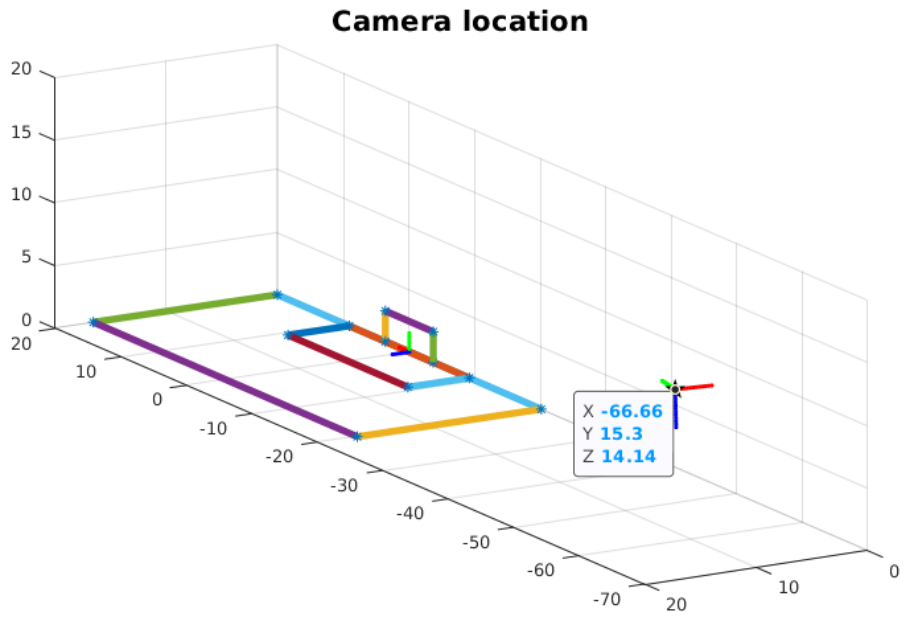
f. Compute the translation vector. Is it reasonable?

$$\mathbf{t} = K^{-1} \cdot M(:,4) \quad \Rightarrow \quad \cdot(-R^{-1}) \quad \mathbf{X}_0 = -(KR)^{-1} \cdot M(:,4)$$

$$\mathbf{t} = -\begin{bmatrix} 2.0717 \\ 13.1499 \\ 68.5584 \end{bmatrix} \ [m] \qquad \mathbf{X}_0 = \begin{bmatrix} -66.6598 \\ 15.3006 \\ 14.1360 \end{bmatrix} \ [m]$$

The results seem reasonable as they satisfy - $\|\mathbf{x}_0\| = \left\|-R^{-1}\mathbf{t}\right\| = \|\mathbf{t}\| = 69.839$ [m], and they express the relative distance of the camera w.r.t the origin (before rotation).

g. Plot the camera location onto the given 3D field model.



Validating the transformation is achieved by plotting **x-y-z** at the origin at **RGB** unit vectors, then performing (Rotation + translation) upon and plotting the transformed X_c.

h. Using all the above information, can you determine if the ball crossed the goal line?

Decision whether the ball has crossed the goal line (namely $X_{y,ball} < 0$) cannot be determined, nonetheless we can describe the intersection line of both planes obtained :

$$\begin{cases} u_{\text{ball}} = \tilde{u} \,./\, \tilde{w} = \mathbf{M}(1,\ :)\mathbf{X}_{3D} \,./\, \mathbf{M}(3,\ :)\mathbf{X}_{3D} \\ v_{\text{ball}} = \tilde{v} \,./\, \tilde{w} = \mathbf{M}(2,\ :)\mathbf{X}_{3D} \,./\, \mathbf{M}(3,\ :)\mathbf{X}_{3D} \end{cases} \quad \text{where} \quad \mathbf{X}_{3D} = [X\ Y\ Z\ 1]^T$$

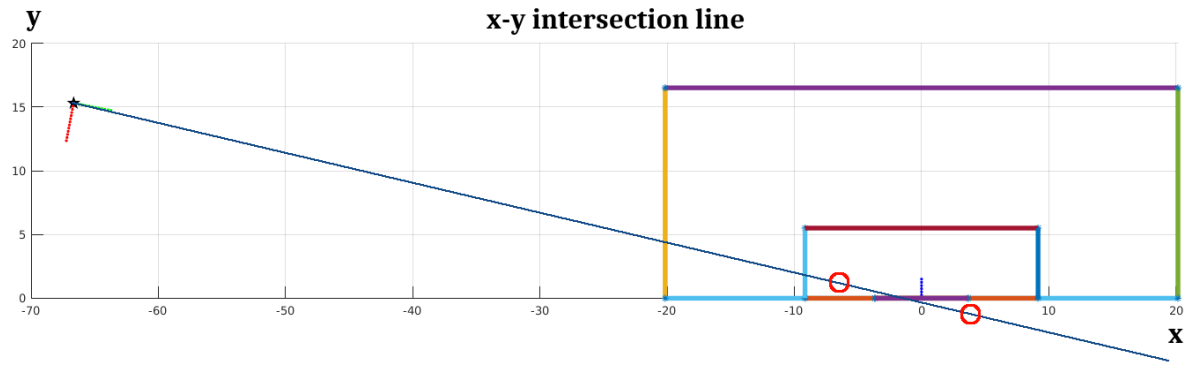
We then get the 2 normalized planes :

$$\begin{cases} z = 14.88x + 64.52y + 18.837 \\ z = -0.1885x + 0.0144y + 1.3504 \end{cases}$$

Which intersect into one line on x-y plane (after normalization) :

$$y = -0.2336x - 0.2711$$

We can plot it as follows :



And now we can see the uncertainty (ambiguity) the image causes. The big question is whether the ball is before the player (not goal), of beyond him (goal).

2 Question

Design and implement an image processing algorithm to automatically determine which of the 5 training images (named leaf{1,2,3,4,5}) is most similar to the testing image (lef6), using **morphological operations** only (of course, other simple image operations are allowed as well).

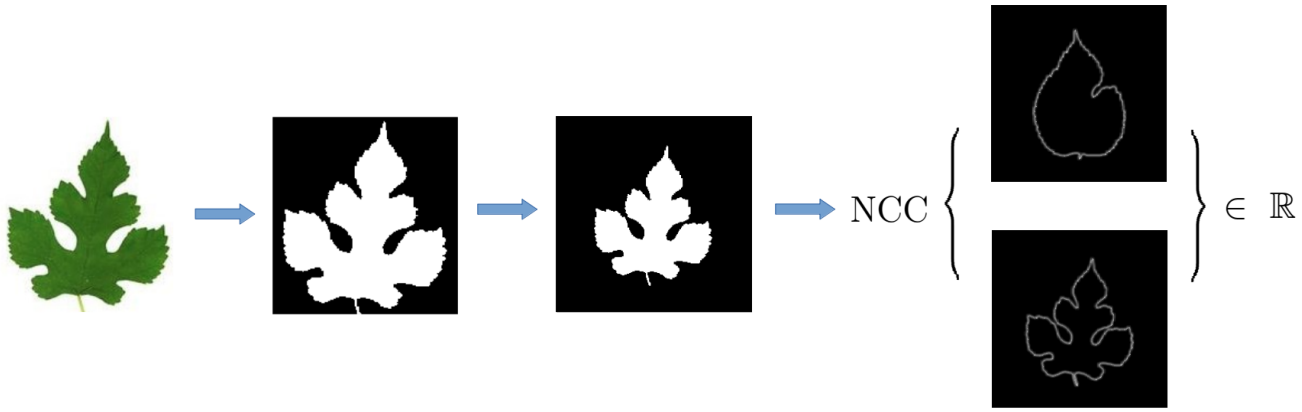
Report the similarity score or distance score between every training image and the testing image, i.e., 5 numbers. Does your algorithm assign the highest similarity score, or equivalently lowest distance score, to the correct training leaf?

The following algorithm works as such :

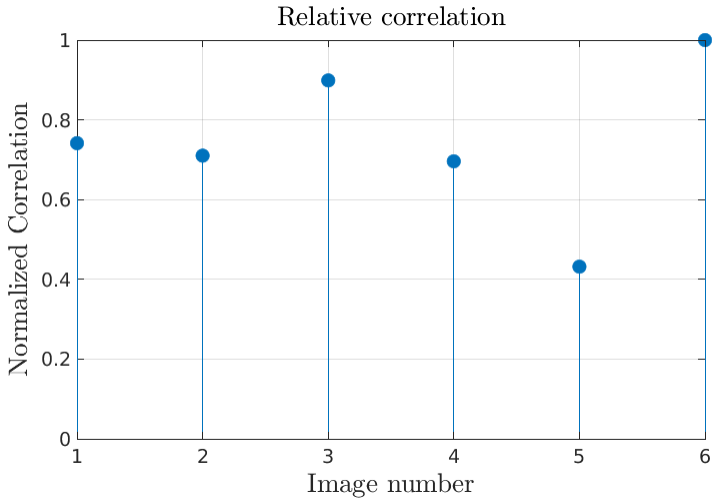
0. Original images (RGB) are sent to algorithm for comparison.
1. Pre-processing stage - pixels under arbitrary grayscale level of (threshold < 165) converted to binary matrix (B/W).
2. Centralize - fit shape into bigger frame such that shape can be extracted fully, and operations between different matrix sizes will be under equivalent conditions.
3. Numerical similarity score - utilizing Normalized Cross-Correlation (NCC) calculation, we would like to match the trained contour shape over the tested one :

$$f_{\text{NCC}}\ {}^{(\text{f-train})}_{(\text{g-test})} = \frac{E\Big((f-\bar{f})(g-\bar{g})\Big)}{\sigma_f \sigma_g} = \frac{\sum_{i,j}(f[i,j]-\bar{f})(g[m-i,n-j]-\bar{g}_{m,n})}{\sqrt{\sum_{i,j}(f[i,j]-\bar{f})^2 \ \sum_{i,j}(g[m-i,n-j]-\bar{g}_{m,n})^2}}$$

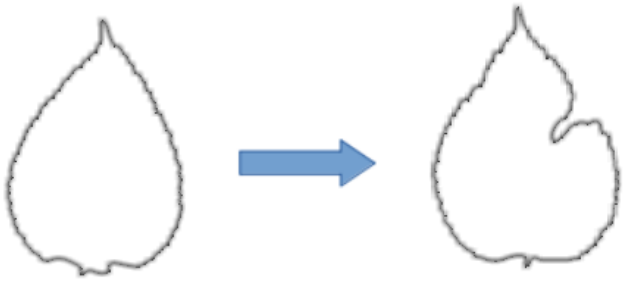
Where m, n are the matrices sizes, and the bar symbolizes mean value. Note that the figures are expressed white on black for the sake of calculations, such that any non-white cell, automatically creates a zero product and thus not effecting the overall computation :



The algorithm plots on-line detected contours (for convenience only), while calculating each step the NCC match between the train / test figures. Eventually, we get the final graph :



The right-most image is the test vs. itself as a control group, and as seen, the third trained image seems to be found most similar to the test figure :



The shape's contours are extracted and matched using NCC

Due to basic and primitive tools learnt so far at course, and under time constraints, the algorithm manages to identify similarity at varying score between [43%, 90%].