DATA.ML.300 Computer Vision

Exercise Round 5

Answered to all questions (1-3)

1)

Let

$$\mathbf{v}' = \mathbf{x}_2' - \mathbf{x}_1' = \begin{pmatrix} x_2' - x_1' \\ y_2' - y_1' \end{pmatrix} = \begin{pmatrix} v_x' \\ v_y' \end{pmatrix}$$

$$\mathbf{v} = \mathbf{x}_2 - \mathbf{x}_1 = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$$

a)

Scale factor can be calculated from the norms of $\mathbf{v}'$ and $\mathbf{v}$. Rotation or translation does not affect to the norm of the vector. Therefore s can be solved from $|\mathbf{v}'| = |s\mathbf{v}|$.

$$\begin{aligned} |\mathbf{v}'| &= |s\mathbf{v}| \\ \sqrt{(x_2' - x_1')^2 + (y_2' - y_1')^2} &= \sqrt{(s(x_2 - x_1))^2 + (s(y_2 - y_1))^2} \\ \sqrt{(x_2' - x_1')^2 + (y_2' - y_1')^2} &= s\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \\ s &= \sqrt{\frac{(x_2' - x_1')^2 + (y_2' - y_1')^2}{(x_2 - x_1)^2 + (y_2 - y_1)^2}} = \sqrt{\frac{{v_x'}^2 + {v_y'}^2}{v_x^2 + v_y^2}} \end{aligned}$$

b)

Cosine of the rotation angle θ can be solved by taking the dot product between normalized $\mathbf{v}'$ and $\mathbf{v}$.

Normed $\mathbf{v}'$: $\mathbf{v}_n' = \frac{1}{\sqrt{{v_x'}^2 + {v_y'}^2}} \begin{pmatrix} v_x' \\ v_y' \end{pmatrix}$

Normed $\mathbf{v}$: $\mathbf{v}_n = \frac{1}{\sqrt{v_x^2 + v_y^2}} \begin{pmatrix} v_x \\ v_y \end{pmatrix}$

$$\begin{aligned} cos(\theta) &= \mathbf{v}_n' \cdot \mathbf{v}_n \\ &= \frac{v_x' v_x}{\sqrt{{v_x'}^2 + {v_y'}^2}\sqrt{v_x^2 + v_y^2}} + \frac{v_y' v_y}{\sqrt{{v_x'}^2 + {v_y'}^2}\sqrt{v_x^2 + v_y^2}} \\ &= \frac{1}{\sqrt{{v_x'}^2 + {v_y'}^2}\sqrt{v_x^2 + v_y^2}}(v_x' v_x + v_y' v_y) \end{aligned}$$

Therefore

$$\begin{aligned} \theta &= cos^{-1}(cos(\theta)) \\ &= cos^{-1}(\frac{1}{\sqrt{{v_x'}^2 + {v_y'}^2}\sqrt{v_x^2 + v_y^2}}(v_x' v_x + v_y' v_y)) \end{aligned}$$

c)

Only the general matrix solution is presented (without setting any results obtained in a) or b)) since otherwise the equations are too messy to give any meaningful insight.

$$\mathbf{x}_1' = sR\mathbf{x}_1 + \mathbf{t}$$
$$\mathbf{t} = \mathbf{x}_1' - sR\mathbf{x}_1$$

Equation opened up a little bit

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \end{pmatrix} - s \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

d)

Let

$$\{\mathbf{x}_1 \to \mathbf{x}_1'\} = \left\{ (\frac{1}{2}, 0) \to (0, 0) \right\}$$

$$\{\mathbf{x}_2 \to \mathbf{x}_2'\} = \left\{ (0, \frac{1}{2}) \to (-1, -1) \right\}.$$

In this situation the parameters are the following

$$s = \sqrt{\frac{{v_x'}^2 + {v_y'}^2}{v_x^2 + v_y^2}} = \sqrt{\frac{(-1-0)^2 + (-1-0)^2}{(0-0.5)^2 + (0.5-0)^2}} = 2$$

$$\theta = cos^{-1}(\frac{1}{\sqrt{{v_x'}^2 + {v_y'}^2}\sqrt{v_x^2 + v_y^2}}(v_x' v_x + v_y' v_y))$$
$$= cos^{-1}(\frac{1}{\sqrt{(-1)^2 + (-1)^2}\sqrt{(-0.5)^2 + (0.5)^2}}((-1)(-0.5) + (-1)(0.5)))$$
$$= cos^{-1}(0)$$
$$= \frac{\pi}{2}$$

$$\begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \end{pmatrix} - \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \end{pmatrix} - 2 \begin{pmatrix} cos(\pi/2) & -sin(\pi/2) \\ sin(\pi/2) & cos(\pi/2) \end{pmatrix} \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$$
$$= \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

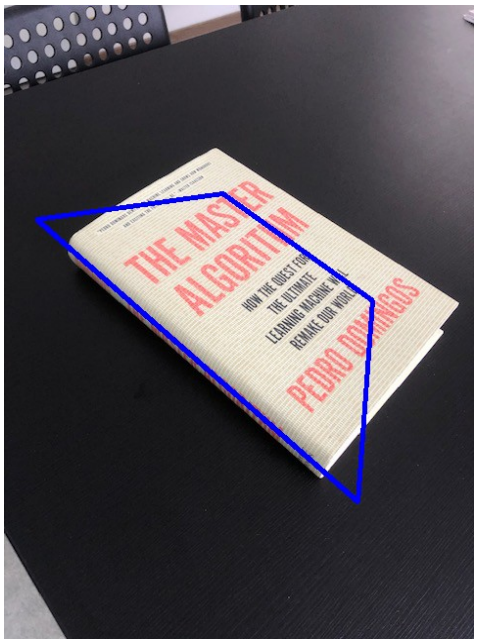
Now the similarity transformation takes the form of

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = 2 \begin{pmatrix} cos(\pi/2) & -sin(\pi/2) \\ sin(\pi/2) & cos(\pi/2) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

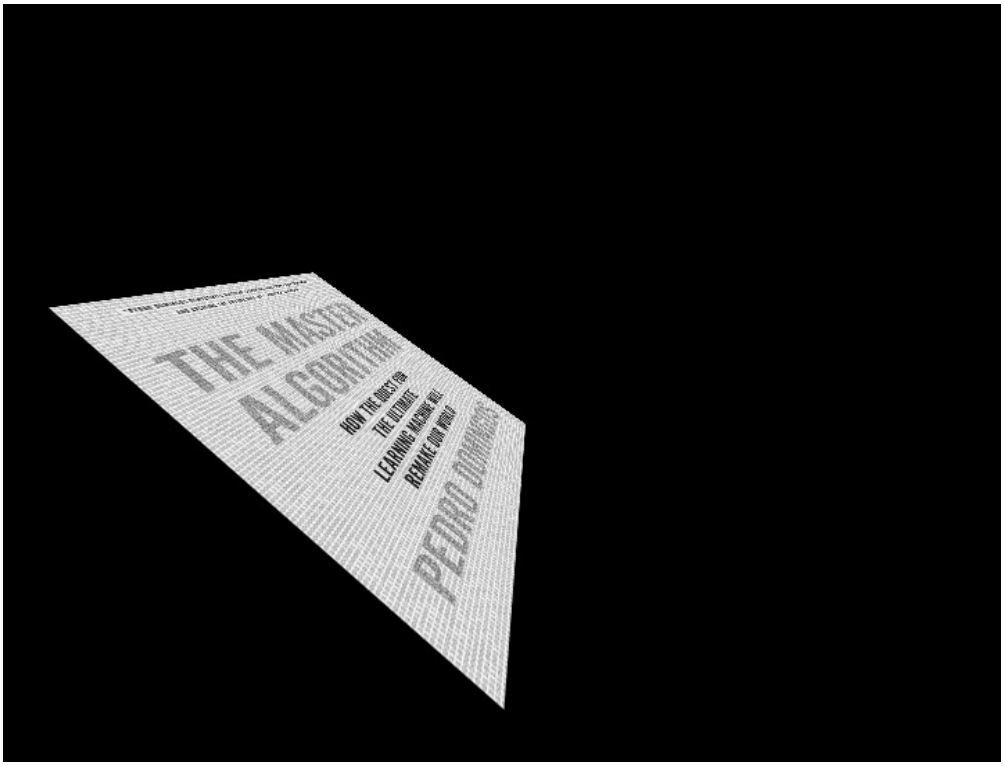
2)

Output from *homography.py*

Homography



Warped image



3)

a)

The main parts of the program are the detection part and the tracking part that can be found from the main loop. In the detection part (if len(p0) <= 10), the program first tries to find the human face and then detect some keypoints from this local area. In the optimal case this procedure have to be performed only a couple of times (or only once). Instead of detecting the face on every frame, the optical flow based tracking part then attempts to estimate the location of the detected keypoints on the next frame (the following else-expression) based on the apparent motion of brightness patterns.

b)

There are some drifting problems with the motion of detected keypoints on subsequent frames. This can be avoided, for example, by using more advanced keypoint detection methods than Harris corner method (eg. SIFT ?) to obtain 'better' keypoints. Therefore the motion of keypoints could be estimated more reliably. Also a some kind of kernel would be used to reduce the noise from the estimated motion (I'm not sure if this is already implemented in opencv's algorithms).

If you have a fast method to detect (correct) keypoints on every frame, this can be also considered. At least in the case of example video (visionface.avi) the raw detection outperformed the optical flow solution without dropping the frame rate.