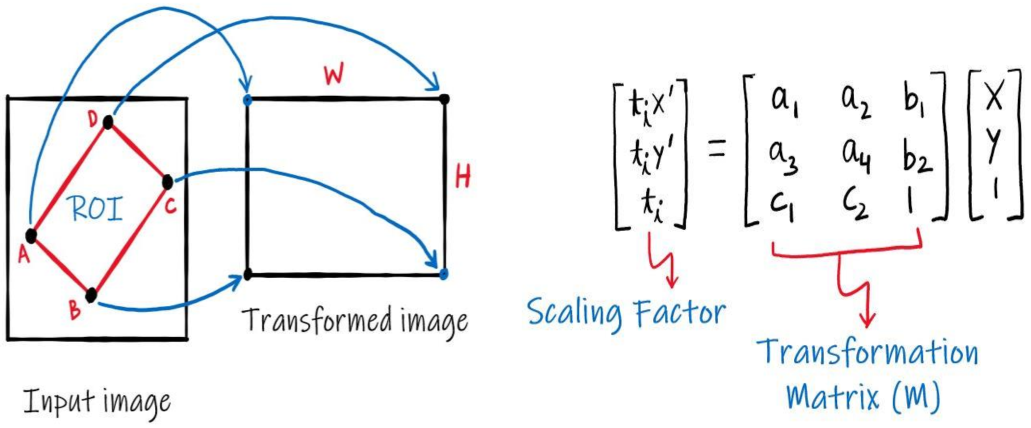
Answers for task02b

1. For this task, notice that the line properties are not preserved but nature is preserved i.e., **lines are still lines.** As we need to project fig. d onto fig. a, b, c, we use projective transform which requires 4 corresponding points for mapping. We consider 4 vertices of fig. c corresponding to the point P, Q, R, S as shown in the manual. Using function **'cv2.getPerspectiveTransform()'**, we have (3,3) matrix, how this works is, if we take any pixel location of one image and multiply it with projective transform matrix then we will get corresponding pixel location of another image (point-to-point mapping). Below figure provides basic understanding of the point made, **of course we go the other way in the lab**. (source:- https://theailearner.com/tag/cv2-getperspectivetransform/)



$$\begin{bmatrix} t_i x' \\ t_i y' \\ t_i \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Suppose **'I'** is original image and **I'** represents image to be transformed then using above ideology, **I'= M . I,** to find '**M**' using matrix manipulation. We have, **M=I'.I**$^{-1}$, this M matrix is what we get using **'cv2.getPerspectiveTransform()'. Either we can find 'M' manually as explain or using 'cv2.getPerspectiveTransform()'.**

2. **We tried** following two methods

   a. Using ffmpeg: change the fps.

   b. Skipping calculations in code and manipulating fps from code. (no edits in video)

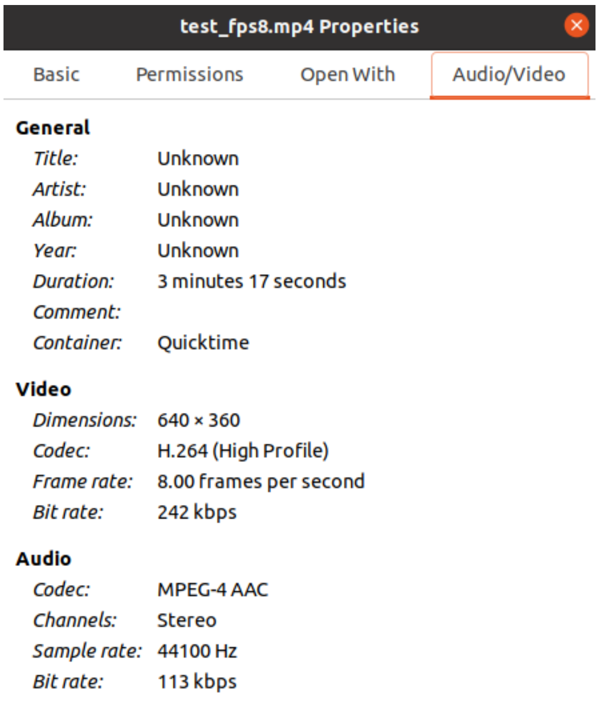
   A. In the first method, we convert the video to a video with different fps.

   We tried several fps like 8,9,10,30 using the following command,

   ffmpeg -i test.mp4 -filter:v fps=fps=8 test_fps8.mp4

   But there was no significant change when it came to running the code, there was more or less the same amount of delay.

Answers for task02b


test_fps8.mp4 Properties

General

Title:	Unknown
Artist:	Unknown
Album:	Unknown
Year:	Unknown
Duration:	3 minutes 17 seconds
Comment:	
Container:	Quicktime

Video

Dimensions:	640 × 360
Codec:	H.264 (High Profile)
Frame rate:	8.00 frames per second
Bit rate:	242 kbps

Audio

Codec:	MPEG-4 AAC
Channels:	Stereo
Sample rate:	44100 Hz
Bit rate:	113 kbps

We suspect because the command preserves the total time which might be resulting in the above delay. Hence we tried another approach which doesn't preserves video's total time,

ffmpeg -i test.mp4 -filter:v "setpts=0.25*PTS" test_2.mp4

This method increases the speed of the video by 4 times. (0.25 in above command)

To run this method while running the file pass command line argument

python3 video_perspect.py --ffmpeg true

Beware, audio is not maintained in these conversions.

B. In second method, simply skipped frames and show 8 frames per second (we tuned the value to be 8)

Basically, we keep a timer and for every (1/8)th second we output one frame.

Support for both methods is given in the code, just run following for Method B.

python3 video_perspect.py

Answers for task02b

3. As given, γ has access to only 3.jpg and arch.jpg, α has access to 1.jpg and 2.jpg, and β has access to 2.jpg and 3.jpg. We tried to establish a relation between 3.jpg and 1.jpg
    a. Let 1.jpg be Img1, 2.jpg be Img2, 3.jpg be Img3
    b. To find relationship between images we can use homography,
        i. Img1 = H12 . Img2
        ii. Img2 = H23 . Img3
    c. So, from above equations,
        i. Img1 = H12 . H23 . Img3
    d. It is implicit that the above transformation is applied for all points in the images.
    e. So, we just need the homography matrices H12 and H23 from α and β respectively and, we will be good.

4. For MSE computation we used traditional method. By considering RGB channel, pixels in image consists of information about colours in (R,G,B) format. We call this information as pixel value. For both the images we have calculated square of difference between corresponding pixel values and then averaged them to calculate **Mean Square Error**. For both ways as explained in question C & D, we got different MSE. Follow is the conclusion regarding this difference:
    a. Because we selected 16 corresponding points in two images, the points aren't greatly coincides with respect to the physical nature of image and hence the difference in MSE.

5. No, 'd' is not same as 'c' as the MSE in both case is different as explained in for the above question. Also the images don't look the same, the image generated with 16 point homography is relatively better than the 4 point perspective transformation.

6. We chose images of television (we also thought of phones and laptops) as we could easily find images of TV from different angles and projecting of img_2.jpg can be done swimmingly.

7. For images we have chosen .png and, for video: container Matroska (.mkv) and codec H.264 (High Profile)