MAEG5720

Computer Vision in Practice

Mini-Project 1

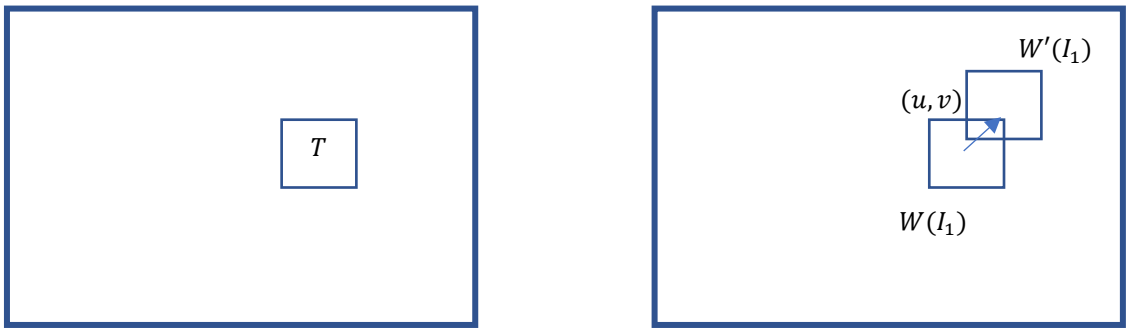
Date: 5th Nov 2021

Due Date: 18:30 26th Nov 2021

Aim: To familiarize with template tracking and detecting moving object in a video stream

In this project, you will implement a simple '**translation-only**' Lucas-Kanade tracker to track a local 2D template in a sequence of image. The Lucas-Kanade works on two consecutive frames each time. We assume the image intensity doesn't change significantly between two frames. The tracking algorithm estimate the deformations between two image frames.

The template tracking algorithm estimate the constant motion (at any image position (x_o, y_o) of the consecutive image frames $I_k \; to \; I_{k+1}$) inside a widow W by inverting the Harris matrix. The motion is expressed as the offset (u, v) that is to be added to the image position (x_o, y_o) and this will be the newly tracked position in image frame I_{k+1}



Given:

(1) The current image frame I_k
(2) The template T in the previous image frame I_{k-1}, centered at image position (x_o, y_o)

Goal: Compute the displacement $\mathbf{p}(u, v)$ using KLT.

Do

1. Warp I with $W(\mathbf{x}; \mathbf{p}))$
2. Subtract I from T $[T(\mathbf{x}) - I(W(\mathbf{x}; \mathbf{p}))]$
3. Compute Gradient $\nabla I(\mathbf{x}')$
4. Evaluate the Jacobian $\frac{\partial W}{\partial \mathrm{p}}$
5. Compute steepest decent $\nabla I \frac{\partial W}{\partial \mathrm{p}}$
6. Compute Inverse Hessian H^{-1}

7. Multiply steepest descend with error $\sum_{\mathrm{x}}\left[\nabla I \frac{\partial W}{\partial \mathrm{p}}\right]^{T}\left[T(\mathbf{x})-I(W(\mathbf{x};\mathbf{p}))\right]$
8. Compute $\Delta\mathbf{p}$

$$\Delta\mathbf{p}=H^{-1}\sum_{\mathrm{x}}\left[\nabla I \frac{\partial W}{\partial \mathrm{p}}\right]^{T}\left[T(\mathbf{x})-I(W(\mathbf{x};\mathbf{p}))\right]$$

9. Update Parameters $\mathbf{p}\leftarrow\mathbf{p}+\Delta\mathbf{p}$

Until $\Delta\mathbf{p}<\in$ **or number of interation** $>\mathbf{n}$

Several Enhancements of the tracking algorithm

(1) As the algorithm assumes very small motion between every two adjacent image frames, any displacement (u,v)>1 pixel cannot be computed. A solution to iterate the same algorithm a number of times is needed. At each iteration step, the window W is shifted by the recovered motion of W'. Then template T and $W'(I)$ is compared (e.g. SSD, or correlation). If the different is larger than the predefined threshold. The iteration will continue until it converges.
(2) We are interested in the centre movement of the window $W$and therefore you may consider giving higher weight to those value of $I_x(x,y)$ and $I_y(x,y)$ near the centre. (e.g. Gaussian)
(3) For large movement, it will be advantages to reduce the resolution of the image (coarse-to-fine approach).

What to do?

You will track a specific object in an image sequence by template tracking. You will apply Lucas-Kanade tracking technique (KLT) to track a template in the image sequence. You will be given a CarSequence.rar file. You will write a MATLAB function to implement the template tracking method and ***translation-only*** model.
In the package, there are a .rar file and two .m files:

(1) *CarSequence.rar* contains the image sequences and the template image.
(2) The script *test_motion.m* is provided to you for testing your function. This script simply (1) repeatedly calls the function *trackTemplate* (that you will code) on every consecutive pair of images in the file *CarSequence.rar*. (2) make an AVI movie out of the *moving_image* returned for every image processed (3) at last save your tracking windows to *track_coordinates.mat*. Please set the *path_to_images* and *numimages* prior to using this function. Please do not submit your AVI file.

(3) The *trackTemplate.m* is the code you will write to track the template in every consecutive pair of images. The script firstly (1) finds the template position in the first image. (2) tracks the template over the rest of the images and (3) at last return the template window coordinates for each image in the image sequence. All image and template will be transformed to gray-scale image for convenience. Four parameters in the *trackTemplate* function

a. Img1 is the template image,
b. Img2 is the image on which you shall track the template
c. Windows 1x4 matrix our the coordinate of top-left and bottom right corners of W. You'll probably notice that the tracker will lose the template before the end of the sequence. To fix this, you may smooth the template and the windows using Gaussian function
d. Sigma is the S.D. of the Gaussian function. Please try the range from 1 to 10.

Summary of codes to compose

Compose the following modules of codes in MATLAB:

(1) *Test_motion.m*: After determining the template position in each of the images, display each color image in the sequence with a bounding box. If your tracker loses the object before the end of the sequence, display a message and exit
(2) *trackTemplate.m*: For the first image, find the initial template window position; for each of the image sequence, iteratively update the window position until the motion (u,v) is small. Then compare the two windows in the current image and the previous image to ensure the error is small enough.

Detail and Hints

(1) Please make a .MAT file which has the top-left and bottom-right corners of your tracked object for evaluating the tracking result. The .MAT file should contain a variable called *TrackedObject* which is an nx4 matrix. Ni is the number of image frames in the input sequence *CarSequence* and each row in the matrix contains 4 numbers: x1, y1, x2, y2 representing the coordinates of the top-left and bottom-right corners of the object you have tracked. Please name the .MAT file as *track_coorinates.mat.*
(2) Since determining the motion vector (u,v) for each frame is an iterative procedure. Please be reminded to set the maximum number of iterations in case your template is lost.
(3) The meshgrid function can be useful for creating array of (x,y) coordinate for all pixels within a car's image windows.
(4) Please vectorize your operations to enhance the running efficiency.

What to submit?

(1) One MATLAB function: trackTemplate
(2) A MAT file: track_coordinates.mat file which contains the track coordinate of car_template.jpg by the trackTemplate function.
(3) A PDF file: miniProject.pdf to describe your template tracking algorithm including the assumptions and parameter choices. (Gaussian sigma value and termination criteria). Please

also include a short discussion about the effect of different smoothing sigma values and possible causes.

Grading Policy:

Successfully track the object which reasonable speed: 70 pts

Discussion on the effect of the sigma value 15 pts

Report Clarity: 15 pts

Additional Bonus: (10 pts)

In the current project, we assume "***translation-only***" model. One can upgrade it into an "***affine***" case by using the warping function

$$W(\mathbf{x};\mathbf{p}) = (a_1x + a_2y + b_1, c_1x + c_2y + b_2)$$

And the Jocobian

$$\frac{\partial W}{\partial \mathbf{p}} = \begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix} \quad \text{and } \nabla I \frac{\partial W}{\partial \mathrm{p}} = [I_x \quad I_y \quad xI_x \quad yI_x \quad xI_y \quad yI_y]$$

You are encouraged to try implement the **"*affine*"** case and shows the result with an appropriate image sequence. 10 pts are given for this upgrade as an encouragement.