EN.601.461/661 – Computer Vision
Fall 2020
Homework #3
Due: 11:59 PM, Monday, December 7, 2020

Please work out both of the problems below in a python notebook. We will supply an autogradable notebook template for you to place your answers into before the due date of the assignment. Data and code to support this assignment will be made available in the class repository.

Motion Field Estimation (20 points)

Recall the motion field equations (note small corrections from the original notes):

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{uT_z - fT_x}{z} + f\omega_y - \omega_z v - \frac{\omega_x uv}{f} + \frac{\omega_y u^2}{f}$$

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{vT_z - fT_y}{z} - f\omega_x + \omega_z u + \frac{\omega_y uv}{f} - \frac{\omega_x v^2}{f}$$

In **`data/mfield1.txt`** and **`data/mfield2.txt`**, we have supplied a list of image coordinates and the corresponding motion vector. Each line "`u, v, du/dt, dv/dt`" represents the image coordinates (u, v) and the corresponding motion vector (du/dt, dv/dt). The focal length is 10 mm for both cases.

With these vectors, do the following:

a. The file **`data/mfield1.txt`** is a motion field for a pure translation in a camera with a focal length of 10mm. Compute the focus of expansion. Hint: the best way to do this is to realize that all lines intersect at a common point, so by writing down the constraint equation for a point to be on the line containing the motion vector, you should be able to solve a linear system to get the solution.

   Print the direction of translation of the camera system.

b. Now, compute the time to collision for each of the points assuming the camera is undergoing pure translation.

c. In **`data/mfield2.txt`**, the camera is both rotating and translating. The rotation motion is (.1, .2, .3) in radians/sec. Correct for this rotation and repeat your calculations of the center of motion, the direction of translation, and the time to collision.

Visual Tracking (40 points)

Given the video sequence `data/tracking.avi`, write code that will track a face (or really any target) through the sequence as it undergoes translation *and* rotation. We will supply the following:

1) An initial bounding box for your tracker to start at.
2) A sequence of images through which a target moves.

You code should output the sequence of locations and orientations of the target.

1) In order to do this, you'll need to extend the image constancy constraint to include rotation. To do this, we will write the image constancy constraint to include translation *and* rotation:

   $$I(R(\theta)\, q + d\ + d_0, t)\ =\ I(q\ +\ d_{0,} t)$$

   where d_0 is the starting location of the template (a constant), q is a location within the tracking window (usually with 0,0 at the middle of the template), d is the offset from the initial starting point, and theta is the change in orientation from the starting template.

   Please calculate and submit in a markdown cell in your python script the following:

   a) Write out the expression above so that each component is explicit, i.e. write it as I(x, y, t) by multiplying q through the rotation matrix and writing an expression for each of the components.
   b) Compute the partial derivative of I with respect to θ in that expression.
   c) Evaluate this expression for $\theta = 0$. This is $dI/d\theta$ evaluated at the template orientation $\theta = 0$.

2) Now that you've done this, you can build a tracking system that operates as follows:

   Initialization:

   1) Subtract the mean of the tracking region from the tracking region (zero-meaning)
   2) Gaussian filter the image; use $\sigma = 2$. Call this T (for template)
   3) Compute the spatial derivatives (gradient) of T
   4) Set up a matrix A with three columns: the first is the x derivatives, the second is the y derivatives, and the last column contains the value of $dI/d\theta$ for each image location
   5) Compute and store $B = (A^tA)^{-1}A^t$

   For tracking, you'll need to do the following:

   1) Given the current d and θ (starting at d = 0 and θ = 0), copy out the portion of the next image in the sequence at d+d0, oriented at θ. *We have provided a Python function for you to do this in the assignment repository.*
   2) Repeat steps 1 and 2 above creating a filtered image -- call it U
   3) Compute the difference between the U and T.

4) Turn the difference into a column vector, and multiply by B
5) Add the result to d and θ
6) Repeat this two or three times
7) Print the result and move on to the next image

In the supplied video tracking.avi, track a region of size 101x101 starting at the location 350, 210.

Extra Credit (5 points): Work this out for scaling as well

Extra Credit (5 points): Make a version that runs directly from your camera.