Video Processing

Lesson 1 – Introduction & Harris Corner Detection

Administration

- Labs would cover the syllabus that is relevant for homework.
- Submission instructions would be published with each exercise.
- Submission should be in pairs (only one should submit), use the forum to find a partner (not needed to be from the same group).
- Late submission penalty is 3pts/day. No need to notify.
- Reception Hours: send an email to schedule a meeting (Email/Skype/Phone).
- For personal issues, please send an e-mail to: danielkigli@mail.tau.ac.il .

More About HW

- **Questions about HW should be asked in the forum on the course site.**
- We will hold ~5 lessons, each of the first three will cover a HW assignment, and the others will be devoted to your final project.
- Lessons will include code, we won't cover the entire code/slides, but it is recommended that you will cover the material (will help with the HW).
- Lessons dates:
  - 25/3 – Harris Corner detector
  - 1/4 – Optical Flow + Video Stabilization
  - Later dates will be given after Passover break

Git

- **Optional if you want.**
- Git is a version control system.
- Why? Useful for version control, not only when working in pairs.
- You can use GitHub (use your student account).

- There are plenty of guides online, use either one you want
  (for example : https://rogerdudler.github.io/git-guide/).

Python

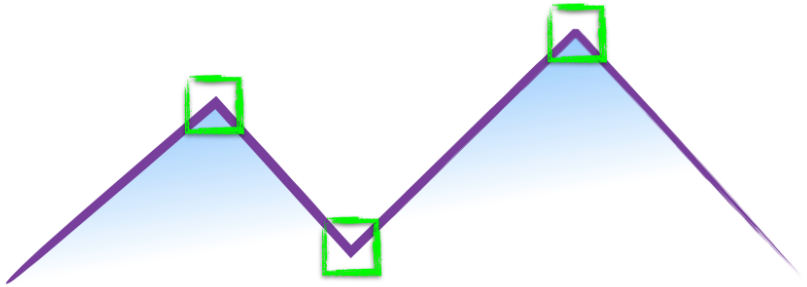
- Lab exercises and HW would be implemented using Python 3.6X .
  - Don’t use Python 2.X
- Main libraries that we will use: numpy, cv2, matplotlib, pillow…
- You may not use a library that wasn’t approved in the HW or taught in class (if such additional library is needed please ask for my approval to use it).
- Assuming you have working knowledge in Python (plenty of guides online).
- Recommend to work in PyCharm (Python IDE).
  - Recommend to create a new virtual env for the course (and install the required packages).
  - A document will be uploaded to the Moodle.

Lab 1 – Python examples

- Numpy and cv2 examples.
- Debugging.
  - Breakpoints (use conditions to find the bug faster).
  - Search for the error, someone has already fixed it..
- Documentation usage.
  - https://docs.scipy.org/doc/numpy/user/quickstart.html
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html
  - https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html
- Find the best question and ask Google, your best friend.

Lab 1 - Harris Corner Detection - Motivation

- What is a corner?



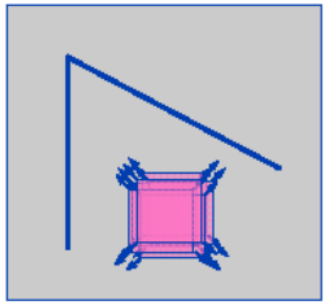
Easily recognized by looking through a small window

Shifting the window should give large change in intensity

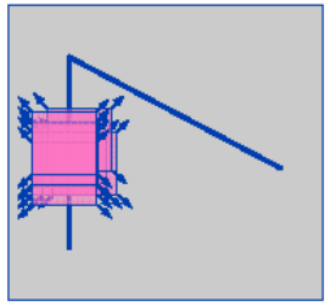
Taken from http://www.cs.cmu.edu/~16385/s17/Slides/6.2_Harris_Corner_Detector.pdf

Lab 1 - Harris Corner Detection - Motivation

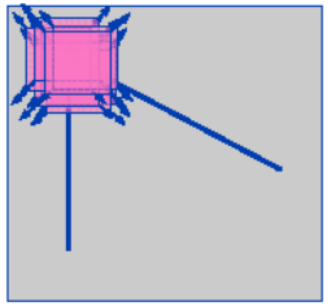
- What is a corner?



"flat" region:
no change in
all directions



"edge":
no change along
the edge direction



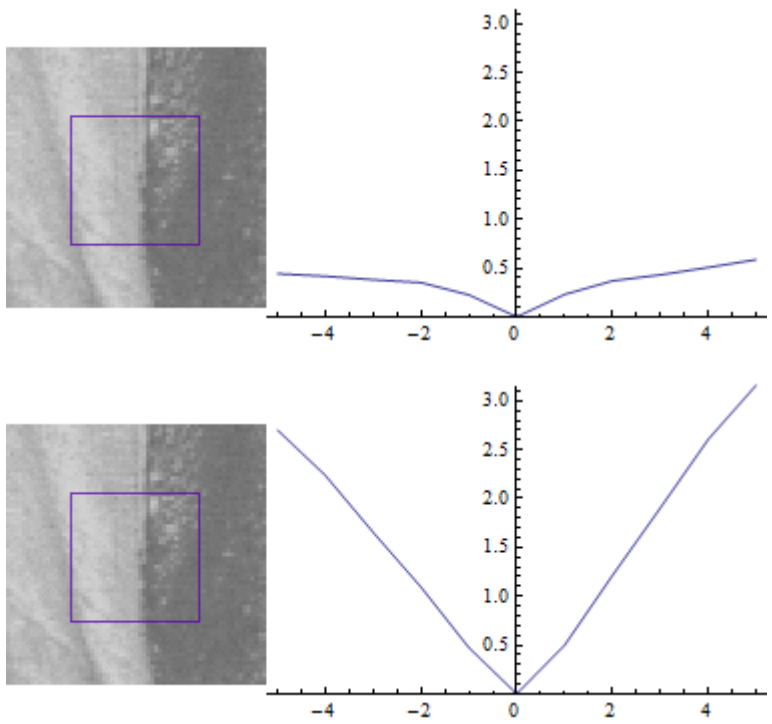
"corner":
significant change
in all directions

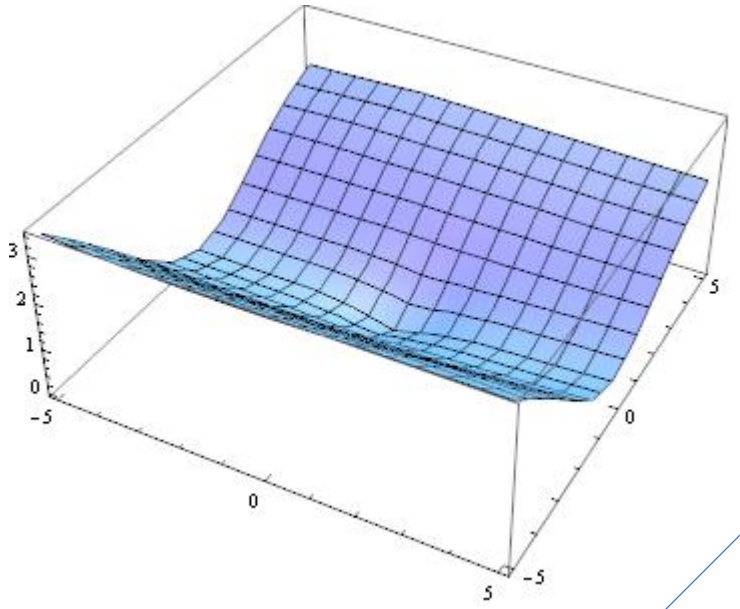
Harris corner detector gives a mathematical approach for determining which case holds.

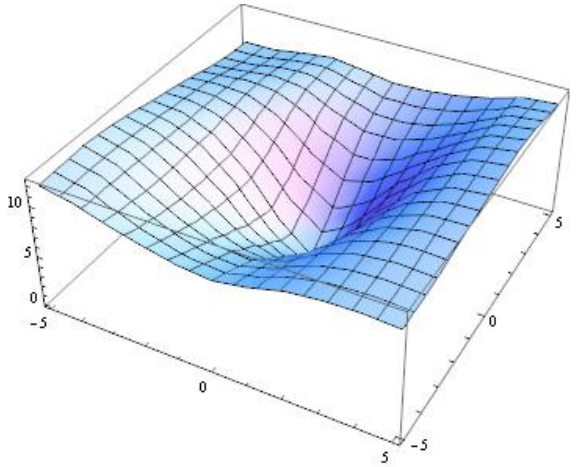
Taken from http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf

Lab 1 - Harris Corner Detection - Motivation

- What is a corner?







Taken from: https://dsp.stackexchange.com/questions/3336/mathematics-of-harris-corner-point-detection

Lab 1 - Harris Corner Detection

$$\sum [I(x+u, y+v) - I(x,y)]^2$$

$$\approx \sum [I(x,y) + uI_x + vI_y - I(x,y)]^2 \quad \text{First order approx}$$

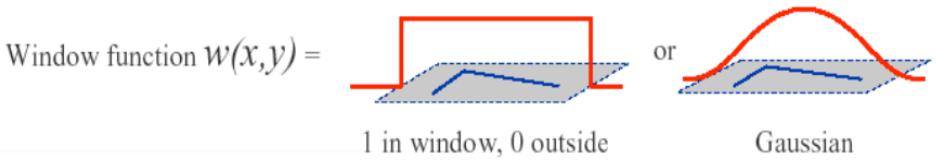
$$= \sum u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2$$

$$= \sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \quad \text{Rewrite as matrix equation}$$

$$= \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Lab 1 - Harris Corner Detection

- We define: $E(u,v) = \sum_{(x,y)\in W}[I(x+u,y+v) - I(x,y)]^2 \approx [u \;\; v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$



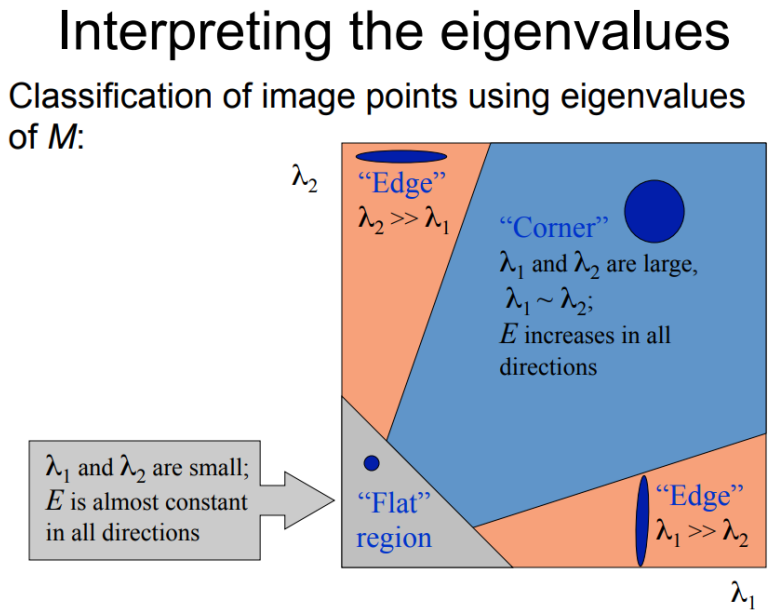
- Paper: http://www.bmva.org/bmvc/1988/avc-88-023.pdf

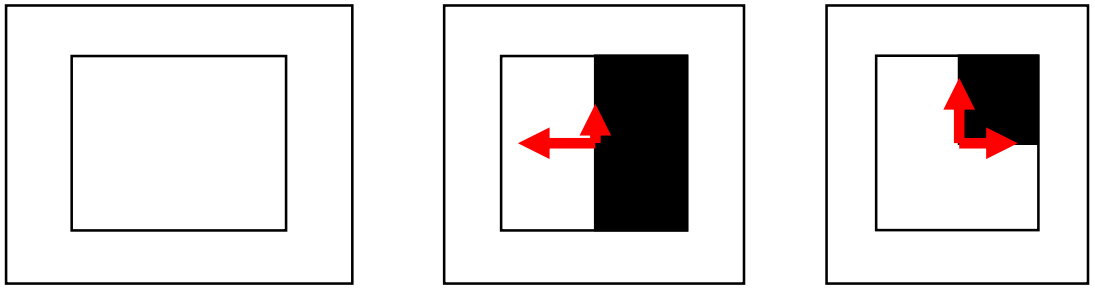
Lab 1 - Harris Corner Detection

- As we saw in class:

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x * I_y \\ I_x * I_y & I_y^2 \end{bmatrix} = A^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} A$$

- Let's compute the eigenvectors of M. What would be their value in each of those cases? What does it mean?

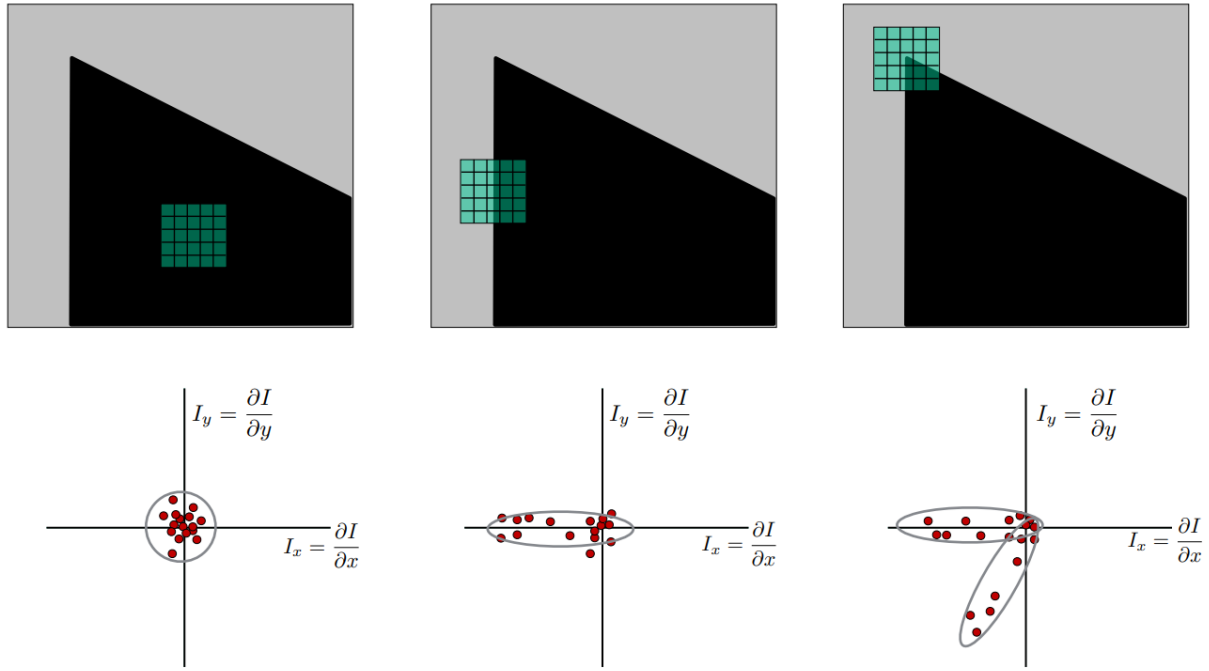




Lab 1 - Harris Corner Detection

- As we saw in class:

$$M = \sum_{x,y} \begin{bmatrix} I_x^2 & I_x * I_y \\ I_x * I_y & I_y^2 \end{bmatrix} = A^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} A$$



Taken from:
http://www.cs.cmu.edu/~16385/s17/Slides/6.2_Harris_Corner_Detector.pdf

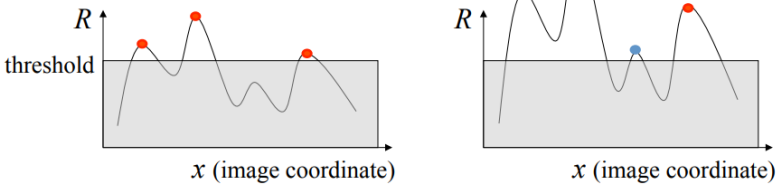
Lab 1 - Harris Corner Detection - Transformations

Affine intensity change

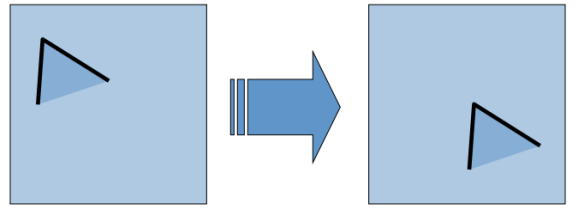
$I \rightarrow a I + b$

Only derivatives => invariance to intensity shift $I \rightarrow I + b$

Intensity scaling: $I \rightarrow a I$

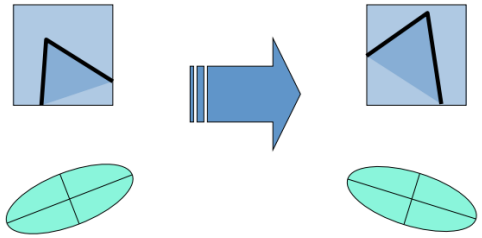


Harris: image translation



Are derivatives and window function shift invariants?

Harris: image rotation



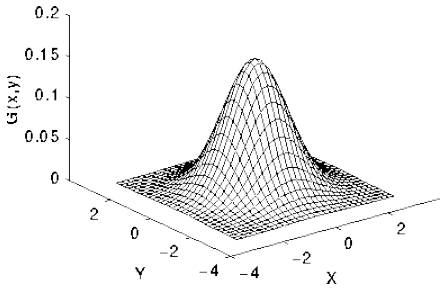
Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Taken from:
http://www.cs.cornell.edu/courses/cs4670/2015sp/lectures/lec07_harris_web.pdf

Lab 1 - Harris Corner Detection- Algorithm

- $[I_x, I_y] = gradient(I)$
  - I_x^2 - pixel-wise multiplication of I_x
- Define filter g – usually box filter (5X5 ones), or gaussian.
- $S_{xx} = conv(I_x^2, \text{g})\ , S_{yy} = conv(I_y^2, \text{g})\ , S_{xy} = conv(I_x \cdot I_y, \text{g})$
- R = $\frac{\lambda_- \cdot \lambda_+}{\lambda_- + \lambda_+} \approx \det(M) - k \cdot [trace(M)]^2 = S_{xx} \cdot S_{yy} - S_{xy}^2 - k\left(S_{xx} + S_{yy}\right)^2$
- $R(R < \theta) = 0$, *where* θ is a user defined threshold, R − Response image
- Optional: Non-maximum suppression of R in each tile

Gaussian filter g



All arrays have the same size as the input image (w,h)

$$Det(M) = \lambda_- \cdot \lambda_+$$
$$Trace(M) = \lambda_- + \lambda_+$$
$$0.04 < k < 0.06$$

Lab 1 – Dividing image into a grid

- You may draw an inspiration from the web (for example https://stackoverflow.com/questions/16873441/form-a-big-2d-array-from-multiple-smaller-2d-arrays).
- The idea is to divide the response image (2d image) into a grid, and in each tile of the grid, return only the maximal value in the tile.
- Afterwards, we'll check if this maximal value is above the threshold (if so, we consider it to be a corner).
- Why? To spread the corners across the image.
- Known as Non-Maximum Suppression.