ENPM673 Project4
Perception of Autonomous Robots
Affine Lucas Kanade Tracker

By (Group 2)
B. Sai Praveen(DirID: spraveen)
Preyash Parikh(DirID: pparikh)
Ajinkya Parwekar(DirID: ajinkyap)

April 20, 2020

1 INTRODUCTION

Optical flow describes a sparse or dense vector field, where a displacement vector is assigned to certain pixel position, that points to where that pixel can be found in another image. In other words, Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera. Lucas Kanade Algorithm is used to find the optical flow of events inside a frame. After considering pixel and adjacent pixel in a frame, the solution is approximated by least square estimation. The change of an object and the pixel are nearly same for two frames and at any moment of time. In order to calculate the flow of motion or vector flow, this same pixel intensity is used between two frames.
The objective of this project is to implement the Lucas-Kanade (LK) template tracker and evaluate the code on given videos namely, featuring a car on the road, a baby fighting a dragon and Usain Bolt's race. The tracker will update an affine frame on each frame and distorts the current so as to make the template in the first frame perfectly aligns with the distorted current frame.

2 LUCAS KANADE ALGORITHM

2.1 LEAST SQUARE CRITERION

'u' and 'v' are the hypothesized location of template in current frame which is given by,

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

$$E(u,v) \approx \sum[I(x,y) + uI_x(x,y) + vI_y(x,y) - T(x,y)]^2$$

$$E(u,v) = \sum[uI_x(x,y) + vI_y(x,y) + D(x,y)]^2$$

By taking partial derivates and equating it to zero:

$$\frac{\partial E}{\partial u} = \sum[uI_x(x,y) + vI_y(x,y) + D(x,y)]I_x(x,y) = 0$$

$$\frac{\partial E}{\partial v} = \sum[uI_x(x,y) + vI_y(x,y) + D(x,y)]I_x(x,y) = 0$$

Form matrix equation which is solved using Least-Squares method:

$$\sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum \begin{bmatrix} I_x D \\ I_y D \end{bmatrix}$$

2.2 LEAST SQUARE - AFFINE PARAMETER

The warp $W(x;p)$ takes the pixel x in the coordinate frame of the template T and maps it to the sub-pixel location $W(x;p)$ in the coordinate frame of the Image I.
$W(x;p)$ denotes the set of parameters.

$$p = (p_1.....p_n)^T \text{ is a vector of parameters.}$$

$$W(x;p) = \begin{pmatrix} x + p_1 \\ y + p_2 \end{pmatrix}$$

The optical flow is now the vector of parameters $p = (p_1, p_2)^T$. To calculate Affine warps,

$$W(x;p) = \begin{pmatrix} (1+p_1)\cdot x + p_3 \cdot y + p_5 \\ p_2 \cdot x + (1+p_4)\cdot y + p_6 \end{pmatrix} = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2.3 MOTIVE OF LUCAS KANADE ALGORITHM

The goal of the Lucas-Kanade algorithm is to minimize the sum of squared error between two images, the template T and the image I warped back to the coordinate frame of the template.

$$X * [I(W(x;p)) - T(x)]^2$$

To compute $I(W(x;p))$ it requires interpolating the image I at the sub-pixel locations $W(x;p)$. Minimization of the above equation is computed with respect to p and the sum is performed all over the pixel x. To optimize the current equation, the current estimate of p is initialized and then iterated over a fixed number to solve the increment for Δp. In this way, the following equation is approximated with respect to Δp and then the parameters are updated.

$$X * [I(W(x;p+\Delta p)) - T(x)]^2$$

The steps are iterated until the estimated p converges.

3 INVERSE COMPOSITIONAL

In Inverse compositional, the paramaters are not calculated in each step but are assumed to be constant for few iterations and then updated. After this, the image and template are swapped.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

The Inverse Compositional algorithm minimizes the following equation with respect to Δp and then updates the warp.

$$X * [T(W(x;\Delta p))I(Wx;p))]^2$$

Here the roles of 'I' and 'T' are now reversed.
The parameters for the inverse affine are as follows.

$$\frac{1}{(1+p_1)\cdot(1+p_4)-p_2\cdot p_3} \begin{pmatrix} -p_1-p_1\cdot p_4+p_2\cdot p_3 \\ -p_2 \\ -p_3 \\ -p_4-p_1\cdot p_4+p_2\cdot p_3 \\ -p_5-p_4\cdot p_5+p_3\cdot p_6 \\ -p_6-p_1\cdot p_6+p_2\cdot p_5 \end{pmatrix}$$

By performing the first order Taylor expansion,

$$\sum_x [T(W(x;0)) + \nabla T \frac{\partial W}{\partial p} \Delta p - I(W(x;p))]^2$$

Assuming W is the identity warp, the least squares question is estimated as,

$$\Delta p = H^{-1} \sum_x [\nabla T \frac{\partial W}{\partial p}]^T [I(W(x;p)) - T(x)]$$

The Hessian matrix where I is replaced with T,

$$H = \sum_x [\nabla T \frac{\partial W}{\partial p}]^T [\nabla T \frac{\partial W}{\partial p}]$$

ALGORITHM - INVERSE COMPOSITIONAL
Pre-Compute:

1) Evaluate the gradient ∇T of the template $T(x)$.
2) Evaluate the jacobian $\frac{\partial W}{\partial p}$ at $(x;0)$.
3) Compute the steepest descent images $\nabla T \frac{\partial W}{\partial p}$.
4) Compute the Hessian matrix.

Iterate:
1) Warp I with $W(x;p)$ to compute $I(W(x;p))$.
2) Compute the error image $I(W(x;p)) - T(x)$.
3) Compute $\sum_x [\nabla T \frac{\partial W}{\partial p}]^T [I(W(x;p)) - T(x)]$.
4) Compute Δp.
5) Update the warp $W(x;p) \leftarrow W(x;p) \circ W(x;\Delta p)^{-1} until ||\Delta p|| \leq \epsilon$

4 HISTOGRAM EQUALIZATION AND PYRAMID

1) Initially, in the car video sequence, due to sudden change in the illumination under the bridge, the tracker breaks down. To overcome this challenge and make algorithm more robust towards illumination, histogam equalization is applied. This helps in equalising the darker intensities.
2)Pyramid technique is used whenever there is a rapid motion. In the pyramid below, there are 'n' layers being shown where the first layer is a normal layer, second and third layer have resolution reduced by two and four times respectively as compared to first layer.

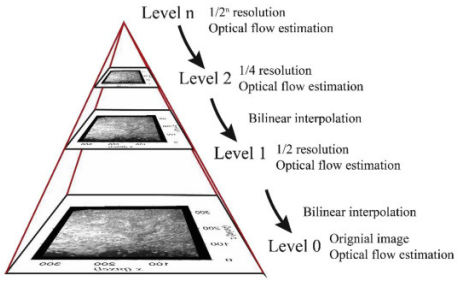

Figure 1: Pyramid Image

In this project, we have implemented pyramid by using inbuilt function "cv2.pyrdown" five times for both current frame and subsequent frames. This was the most suitable results for us.

In the pyramid technique, the parameters are calculated from bottom to top layer and is warped to layer above it with updated parameters. Therefore at the end, the parameters are updated to the first layer by considering the rapid shifts in the lower resolution levels.

PIPELINE
Initially, a template is defined by drawing a rectangle around the object that needs to be tracked. The next frames are passed through Lucas Kanade algorithm which returns Δp. Then the parameters are updated in the current frame $(p + \Delta p)$. After the convergence of error below some threshold value, the parameters are returned and the updated parameters are used to draw the rectangle and is warped back to the current frame.

KEY POINTS
1) LK inverse compositional Algorithm is preferable as it gives more robust solution. Also, in general LK algorithm, cost to calculate Hessian Matrix is significant.
2) Threshold for *Deltap* is about 0.05, 0.005, 0.005 for baby dragon, 0.001 for car and 0.09 for bolt respectively.
3) The algorithm breaks down when there is change in illumination. This is avoided by using histogram.
4) Pyramid technique is used for rapid movement.

5 OUTPUT

TEMPLATE

Figure 2: First frame of car

Figure 3: First frame of baby dragon

Figure 4: First frame of bolt

RESULT
The output images for the car are as follows.

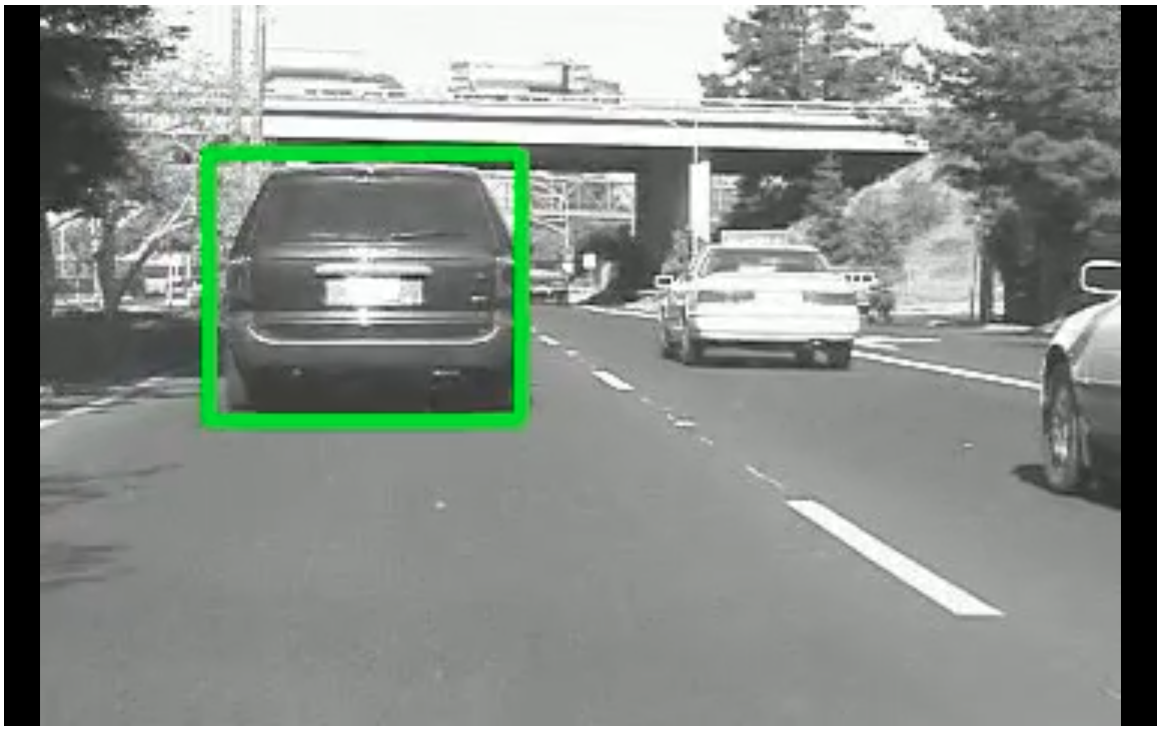

Figure 5: Output image for car-1

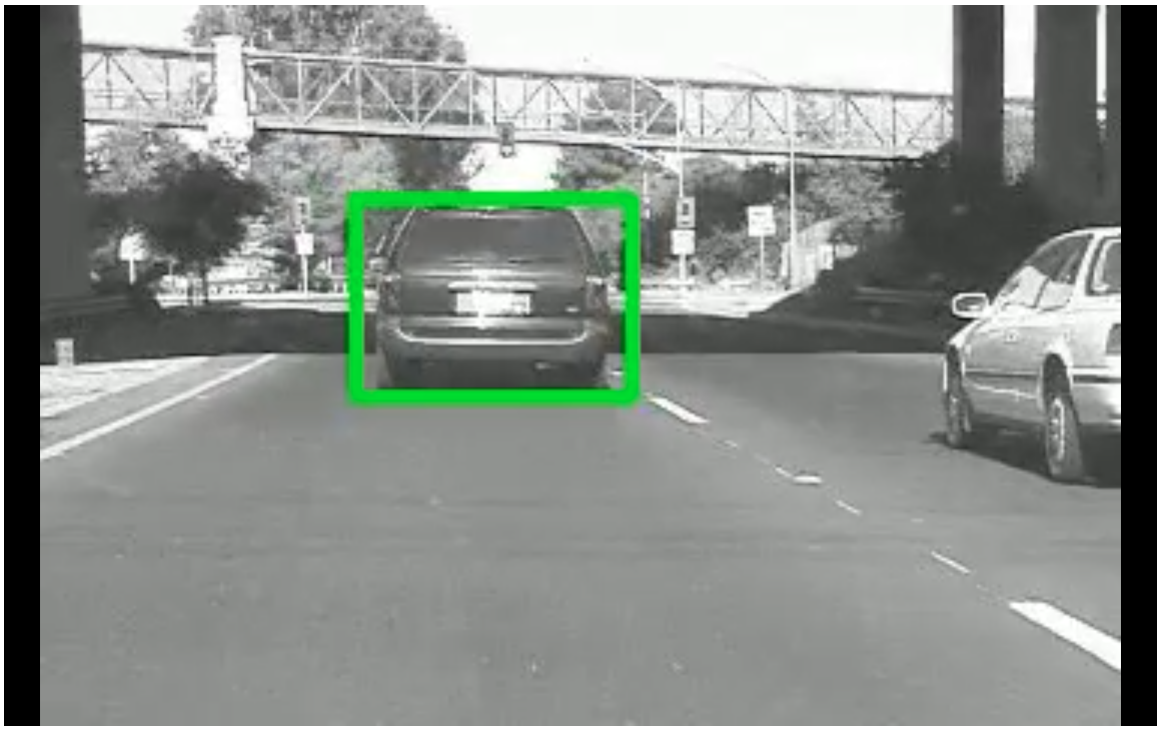

Figure 6: Output image for car-2

Figure 7: Output image for car-3

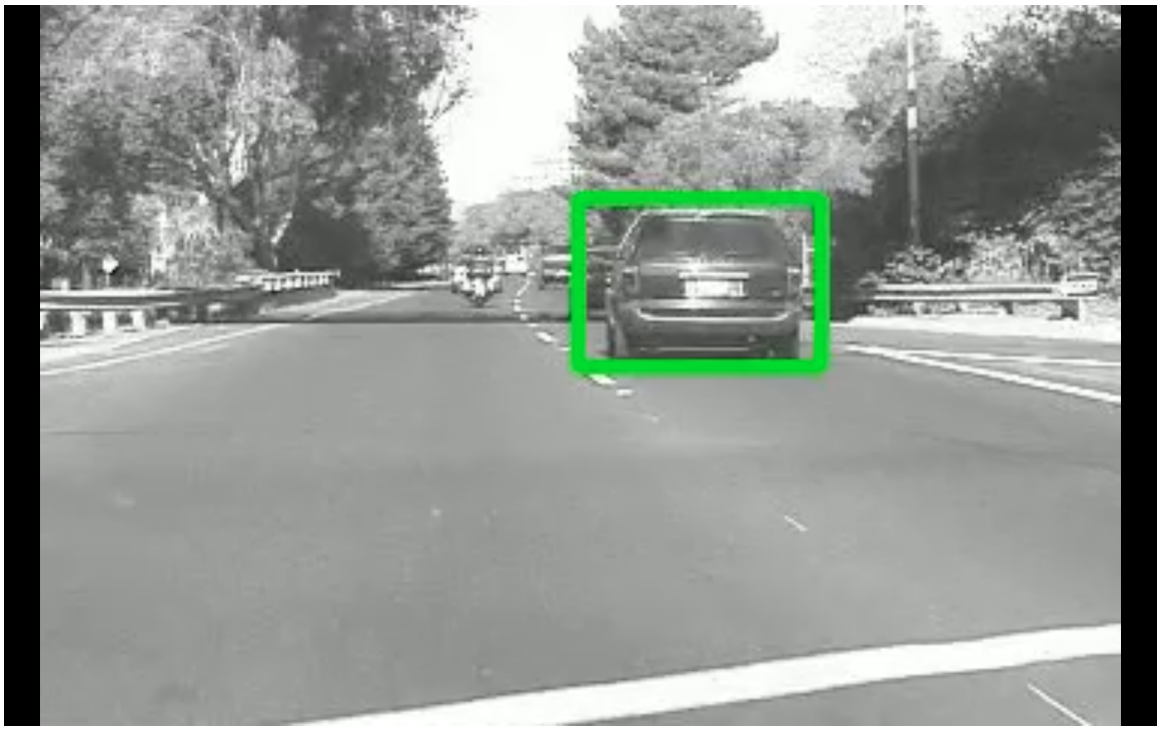

Figure 8: Output image for car-4

The output images for the baby dragon are as follows.

Figure 9: Output image for baby dragon-1

Figure 10: Output image for baby dragon-2

Figure 11: Output image for baby dragon-3

The output images for the bolt are as follows.

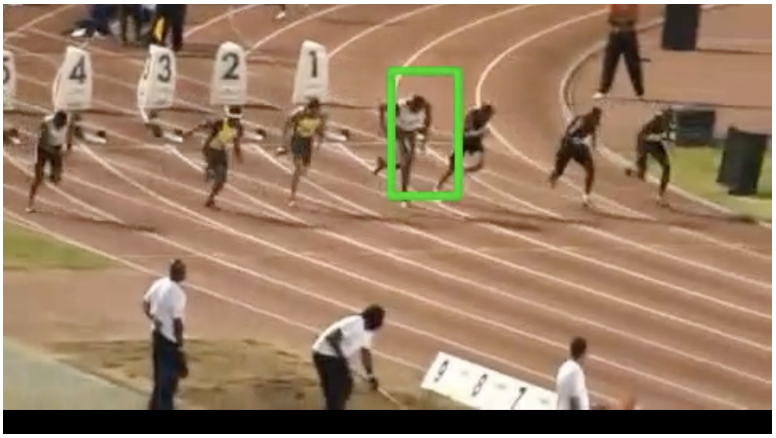

Figure 12: Output image for bolt-1

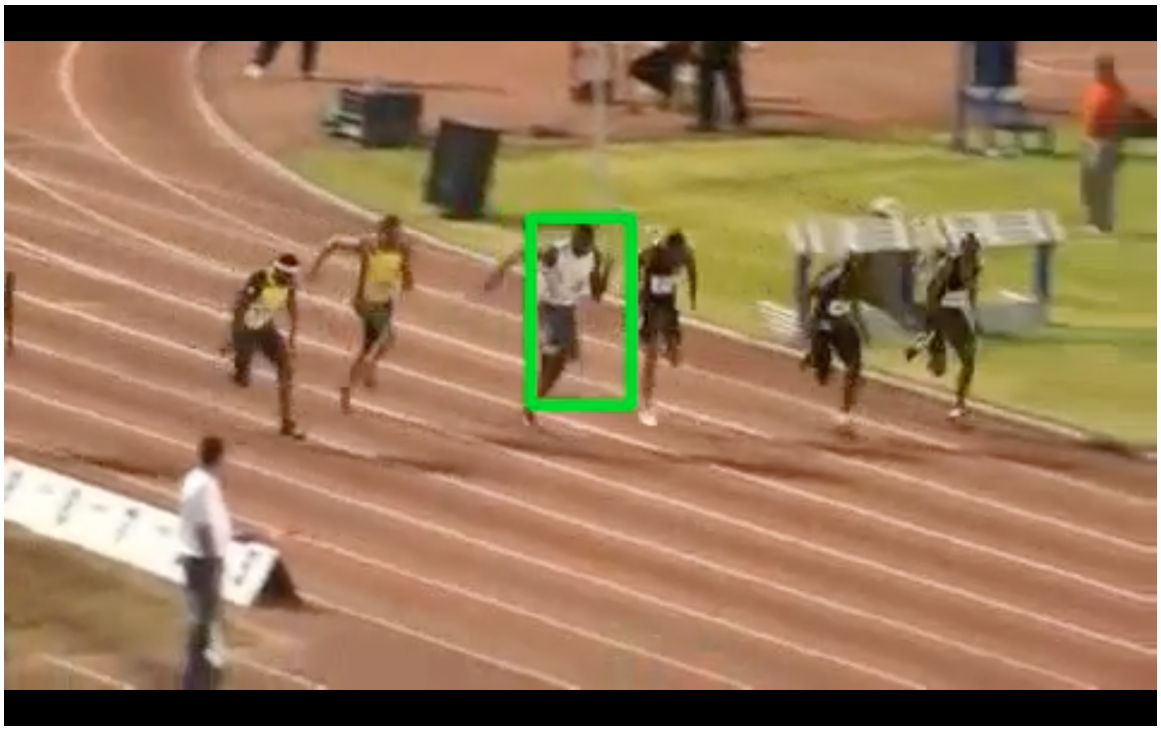

Figure 13: Output image for bolt-2

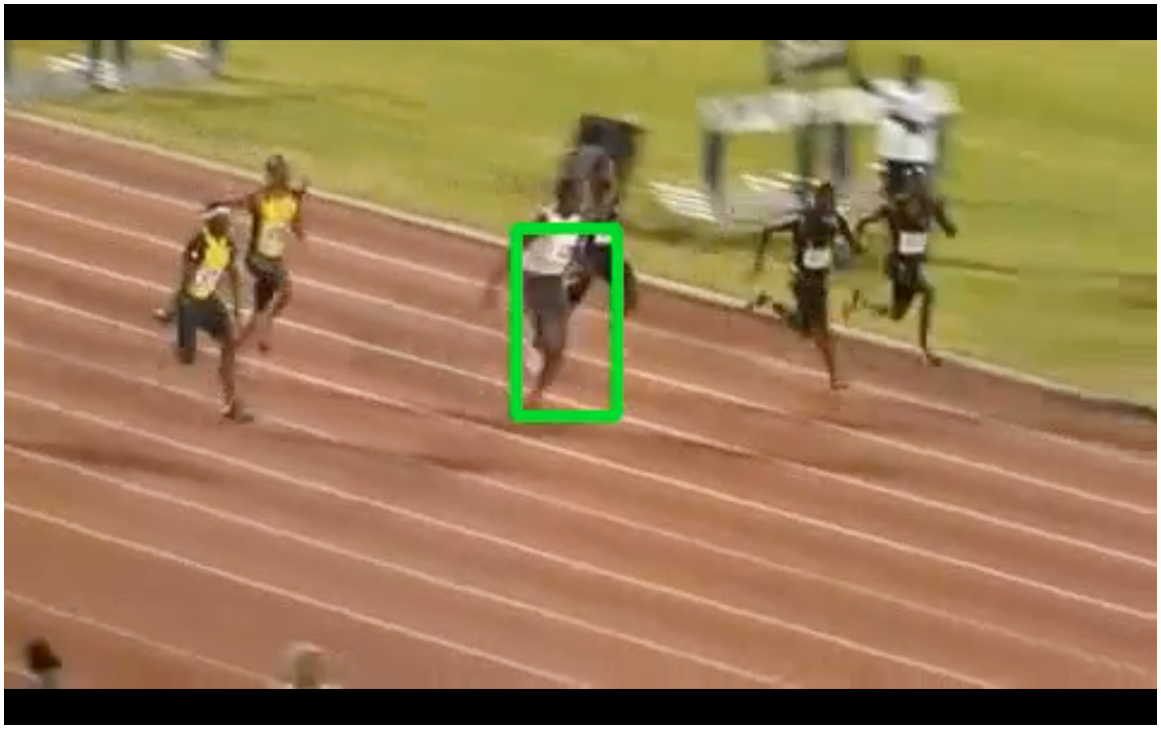

Figure 14: Output image for bolt-3

The below is the google drive link for the final output videos. https://drive.google.com/drive/folders/12Nor1q7n56XeC4IhvihYuUT_PlBZBmLF

6 REFERENCES

1) https://www.ri.cmu.edu/pub_files/pub3/baker_simon_2002_3/baker_simon_2002_3.pdf
2) http://cvlab.hanyang.ac.kr/tracker_benchmark/benchmark_v10.html
3) https://www.youtube.com/watch?v=tzO245uWQxA&list=PLmyoWnoyCKo8epWKGHAm4m_SyzoYhslk5&index=10
4) https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=pyrdown#pyrdown
5) https://www.pyimagesearch.com/2015/10/05/opencv-gamma-correction/
6) http://xahlee.info/img/what_is_gamma_correction.html
7) ENPM 673 course material.