**《Computer Vision Foundations and Applications》Homework 4**

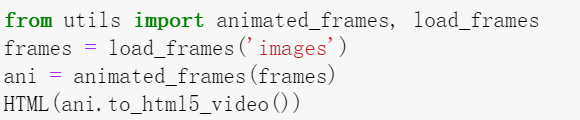
**Attention: Please follow the instruction in hw4.ipynb file and finish it with all the outputs and your answers to the written questions.**

This assignment covers **Lukas-Kanade tracking method**.

## 0. Displaying Video

We have done some cool stuff with static images in past assignemnts. Now, let's turn our attention to videos! For this assignment, the videos are provided as time series of images. We also provide utility functions to load the image frames and visualize them as a short video clip.

Note: You may need to install video codec like *[FFmpeg](http://ffmpeg.zeranoe.com/builds/" \t "_blank)*. For Linux/Mac, you will be able to install ffmpeg using *apt-get* or *brew*. For Windows, you can find the installation instructions [*here*](https://www.wikihow.com/Install-FFmpeg-on-Windows).



## 1. Lucas-Kanade Method for Optical Flow

### 1.1 Deriving optical flow equation

You might be wondering how we went from the brightness constancy assumption to the optical flow equation. Try to derive it yourself!

* **a.** Derive the optical flow equation from the brightness constancy equation. Clearly state any assumption you make during derivation.
* **b.** Can the optical flow equation be solved given two consecutive frames without further assumption? Which values can be computed directly given two consecutive frames? Which values cannot be computed without additional information?

**Your answer1: Write your answer in hw4.ipynb’s 1.1 markdown cell. (in Chinese)**

### 1.2 Overview of Lucas-Kanade method

* **a.** What is the condition for this equation to be solvable?
* **b.** Reason about why Harris corners might be good features to track using Lucas-Kanade method.

**Your answer2: Write your answer in hw4.ipynb’s 1.2 markdown cell. (in Chinese)**

### 1.3 Implementation of Lucas-Kanade method

Implement1 function **lucas\_kanade** in motion.py and run the code cell below. You will be able to see small arrows pointing towards the directions where keypoints are moving.

### 1.4 Feature Tracking in multiple frames

Implement2 **compute\_error** in motion.py, and re-run the code cell below. You will see many of the points disappearing in later frames.

*Note: The output needs to include the results of the first run and the second run.*

## Pyramidal Lucas-Kanade Feature Tracker

### 2.1 Iterative Lucas-Kanade method

Implement3 **iterative\_lucas\_kanade** method in motion.py and run the code cell below. You should be able to see slightly longer arrows in the visualization.

### 2.2 Coarse-to-Fine Optical Flow

Implement4 **pyramid\_lucas\_kanade** method in motion.py .

## Object Tracking

### 3.1 Evaluating Object Tracker: intersection over union (IoU)

Intersection over union is a common metric for evaluating performance of an object tracker. Implement5 **IoU** in motion.py to evaluate our object tracker. With default parameters, you will get IoU score of ~0.32.

## Extra Credit: Optimizing Object Tracker

Optimize object tracker in the code cell below. You may modify the code and define new parameters. We will grant extra credit for IoU score > 0.45.

***Attention:* *The total score will not exceed 100 with Extra Credit.***