ENPM673 - Perception for Autonomous Robots

Project 4

Prateek Arora, Abhinav Modi, Samer Charifa

Due date: *20th April 2020, 11:59 p.m.*

Submission Guidelines

- The homework is to be completed in group and there should be a single submission per group.
- Your submission on ELMS/Canvas must be a zip file, following the naming convention **YourDirectoryID_proj_4.zip**. If you email ID is abc@umd.edu or abc@terpmail.umd.edu, then your DirectoryID is **abc**. Remember, this is your directory ID and **NOT** your UID. Please provide detailed instructions on how to run your code in README.md file.
- For each section of the homework, explain briefly what you did, and describe any interesting problems you encountered and/or solutions you implemented. Your report should preferable be typeset in LaTeX.
- Please submit only the python scipt(s) you used to compute the results, the PDF report you generate for the project and a detailed README file (refer to the directory structure below).
- Include sample outputs in the your report.
- The video outputs are to submitted as a separate link in the report itself. The link can be a YouTube video or a google drive link (or any other cloud storage). Make sure that you provide proper sharing permission to access the files. **If we are not able to access the output files you will be awarded 0 for that part of the project**. Provide video output for all three dataset.
- Please don't include the dataset provided to you and your video results in your submission, rather provide the link to your video output in the report.
- Disallowed function:
  - any inbuilt function from scipy or sklean that directly implements a tracker.
  - any other inbuilt function that solves the question in less than 5 lines.

The file tree of your submission SHOULD resemble this:

```
YourDirectoryID_hw1.zip
├── Code
│   ├── .py files
│   └── any subdirectories that you may have
├── Report.pdf
└── README.md
```

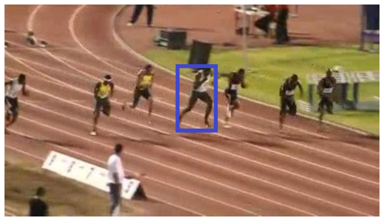
Data

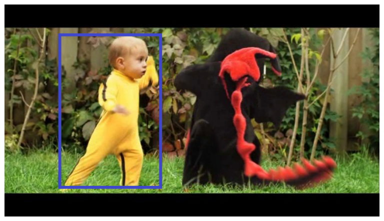
The project requires you to track three different objects in three different videos which can be found here:

1. Dataset 1
2. Dataset 2
3. Dataset 3

Introduction

In this project you will implement the Lucas-Kanade (LK) template tracker. Then you will evaluate your code on three video sequences from the Visual Tracker benchmark database: featuring a car on the road, a baby fighting a dragon, and Usain bolt's race. The short video that you will process are available in the data section.
To initialize the tracker you need to define a template by drawing a bounding box around the object to be tracked in the first frame of the video. For each of the subsequent frames the tracker will update an affine transform that warps the current frame so that the template in the first frame is aligned with the warped current frame. For extra credit, you may look at ways to make tracking more robust.


(a) Bolt


(b) Car

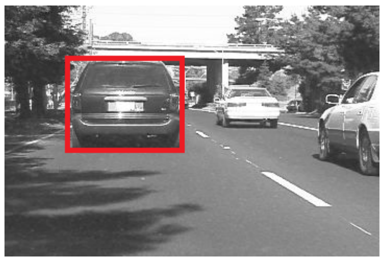

(c) DragonBaby

Figure 1: Tracking sequences

Lucas Kanade Template tracker **[80points]**

Implementation of the Tracker

Technical background and the implementation of the LK tracking algorithm are described in Section 2 of Simon and Matthew's paper. Initialize manually the coordinates of a bounding box surrounding the object to be tracked. You have to define an initial bounding box for each video and provide the chosen template along with your submission. At the core of your algorithm is a function affineLKtracker(img, tmp, rect, p_{prev}). This function will get as input a grayscale image of the current frame (img), the template image (tmp), the bounding box (rect) that marks the template region in tmp, and the parameters pprev of the previous warping. The function will iteratively compute the new warping parameters p_{new}, and return these parameters. You can either use a fixed number of gradient descent iterations or formulate a stopping criteria for the method. Your algorithm should compute the affine transformations from the template to every frame in the sequence and draw the bounding boxes of the rectangles warped from the first frame.

Evaluation of the Tracker

Evaluate your tracker on the three sequences: the car sequence, the bolt sequence, and the baby fighting a dragon. Use only the grayscale, not the color. What sort of templates work well for tracking? At what point does the tracker break down? Why does this happen?

Robustness to Illumination **[10 + 10 points]**

The LK tracker as it is formulated, breaks down when there is a change in illumination because the sum of squared distances error that it tries to minimize is sensitive to illumination changes. There are a few things you could try to do to fix this. The first is to scale the brightness of pixels in each frame so that the average brightness of pixels in the tracked region stays the same as the average brightness of pixels in the template.

The second is to use a more robust M-estimator instead of least squares (e.g. a Huber loss) that does not let outliers adversely affect the cost function evaluation. Note that doing this will modify the least squares problem to a weighted least squares problem, i.e. for each residual term you will have a corresponding weight Λ_{ii} and your minimization function will look like

$$L = \sum_{x} \Lambda_{ii} \left[T(W(x;\Delta p)) - I(W(x;p))\right]^2 \tag{1}$$

leading your Jacobian computation to be a weighted Jacobian

$$A^T \Lambda A \Delta p = A^T \Lambda b \tag{2}$$

Thus,

$$\Delta p = (A^T \Lambda A)^{-1} A^T \Lambda b \tag{3}$$

Here Λ is a diagonal matrix of weights corresponding to the residual term computed as per the choice of the robust M-estimator used, A is the Jacobian matrix for each pixel of the template considered in the cost function, and b is the corresponding vector of residuals. Implement these two methods and test your new tracker on the car video sequence again.

Credits: This project has been inspired by homework assignment 7 from CMU's Computer Vision course in Spring 2018