Computer Vision Project (CS 600.361/600.461)

Instructor: Nicolas Padoy (padoy@jhu.edu)
TA: Maria Ayad (maria.ayad@jhu.edu)

Out on Thursday, October 20, 2011

Schedule

1. Oct. 28, 11:59pm - Short description

   A one or two page description is due, containing: 1) Team members; 2) Project's choice; 3) Proposed approach and final functionality; 4) Project components and team member assignments (tentative); and 5) Rough project timeline.

2. Nov. 14, 11:59pm - Progress report

   By this time, you should have some code running. Please submit an update to your one-page description indicating what you've accomplished, and what (if any) problems you've run into. For project #1, include pictures of the object you try to reconstruct.

3. Nov. 29 / Dec 1 - Short oral presentation

   You have a 10 minutes slot in class to present your approach and the current results (using a laptop and a few slides).

4. Dec 2, 11:59pm - Code and final report

   Submit a single .zip file named with your jheds using the format **jhed1_jhed2_jhed3_project.zip**. This file should contain within a directory named **jhed1_jhed2_jhed3_project** your **commented** matlab code, your data, your final video and your final report (including your names) describing your approach and results (.pdf or .doc file).

Evaluation.
Your solution will be evaluated with respect to 1) completeness of the project; 2) clarity of the writeup and 3) interest/complexity of the methods implemented. See each project for more details.

Integrity code / Allowed libraries.
Projects can be carried out in groups of two or three students. Any code used or provided for the homework can be used for the project. Additionally, the library VLFeat[1] (containing e.g. code for computing SIFT features), Piotr's toolbox[2] (containing e.g. code for computing optical flow), libSVM[3] (containing an implementation of SVM classifiers), the HMM/Kalman toolboxes by Kevin Murphy[4] and the Matlab camera calibration toolbox[5] (**ONLY** to calibrate the internal

[1] `http://www.vlfeat.org/`
[2] `http://vision.ucsd.edu/~pdollar/toolbox/doc/`
[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
[4] `http://www.cs.ubc.ca/~murphyk/Software/`
[5] `http://www.vision.caltech.edu/bouguetj/calib_doc/`

parameters of a camera with the toolbox's gui) can be used if needed. Please also note that by submitting your solution, you abide by the CS department Academic Integrity Code.

Projects. You can choose between the two following projects. The descriptions below are indicative only: you are strongly invited to be creative and to propose your own solutions and improvements.

1. **Reconstruction of a Hopkins landmark**

   The objective of this project is to perform a 3D point cloud reconstruction of a Hopkins landmark using a single camera. Here are a few exemplary steps to obtain such a reconstruction:

   (a) Calibrate your camera, e.g. using the Matlab camera calibration toolbox and a calibration pattern.

   (b) Select a landmark on the campus that you are interested in reconstructing. This landmark can have any size, but should contain depth differences, features and textured parts in order to be interesting and feasible to reconstruct. Take two shots (more if you wish) of the object from different camera locations.

   (c) Propose a solution to compute the relative motion (R,T) between the two camera positions where the images have been taken. Different possibilities exist, involving a different amount of manual interactions or scene knowledge. For instance, the relative camera motion can be obtained using 4 corner points from a planar marker inserted in the 3D scene and seen in the two images (compare this to the planar camera calibration approach), or by extracting R and T from the essential matrix, up to a scale factor (such an approach is described in the books HZ 9.6.2 or RS 7.2).

   (d) Perform a sparse point reconstruction by matching (use the epipolar constraint) and triangulating Harris or SIFT features. Alternatively, a dense reconstruction can e.g. be obtained using stereo rectification and dense feature matching.

   (e) Map colors to the 3D points and present your results by showing the 3D reconstructed point cloud under different views (ideally in a movie showing the reconstruction under different orientations -check the Matlab function *view*).

   Since there exists different approaches with different advantages/drawbacks for all the steps described above, choose the one that you prefer, but try to minimize the required amount of manual interactions by using techniques such as RANSAC to estimate your models and correspondences. You are also encouraged to improve your reconstruction by using more than two images.

   The baseline is an approach that yields an imperfect but recognizable sparse reconstruction of the object using images from a single camera. **In 600.461**, you need to show results demonstrating your efforts towards a *dense* reconstruction.

2. **Object detection and classification in a video sequence**

   The purpose of this project is to detect and recognize the moving objects (cars, yellow taxis, motorbikes, bicycles, pedestrians) present in the video *traffic.avi*. Exemplary steps for the project are described below.

(a) Detect the different moving objects (blobs) in the video, using for instance techniques based on optical flow or background subtraction. If needed, you can stabilize the camera motion by registering affinely the frames using SIFT features. Try to enforce spatial and temporal consistency during the detection.

(b) Implement feature vectors that characterize the different objects, based for instance on their appearance and motions.

(c) Generate training data using the beginning of the video. (The last 25 seconds can only be used for evaluating your approach.)

(d) Implement a classification method to identify the moving objects. Try to take the differences in scale and shapes into account.

(e) Generate a video showing your results by identifying in each frame each detected object by a bounding box. Each bounding box should be identified by a distinctive number (to follow the same object across frames) and have a color corresponding to the object class.

As for the previous project, different approaches exist to detect the blobs, generate feature vectors and to classify/recognize the objects. Choose approaches that you are interested in; if you try out different approaches, you are encouraged to present and compare the results from the different approaches in your report.

The baseline is an approach that yields an imperfect but visually acceptable detection of the moving cars and yellow taxis. **In 600.461**, you need to show results demonstrating your efforts towards the detection and recognition of the remaining classes (motorbikes, bicycles, pedestrians).

Report format
Your final report should document in max. 6 pages the following:

1. Introduction

   Describe the goals of your project.

2. Methods

   Present the computer vision methods that you used to achieve these goals. Explain where manual input is required in your approach.

3. Results

   Describe and show the goals/results you achieved.

4. Conclusions

   Give the known limitations of your method and possible future extensions.

5. Code information

   Present what needs to be known in order to run your code and indicate which libraries are required as well as what they are used for.

You are encouraged to use LaTeX[6] to write your report.

Grading
Note: you are allowed to read as much computer vision literature as you want to do the project. But aside from the allowed libraries, you need to implement the code for the project yourself.

Tentative grading scheme:

- Implementation (commented code) 30%
- Results (video) 15%
- Presentation: 15%
- Intermediate report: 10%
- Report: 30%

Groups
The project can be performed by groups of two or three students. If one student of the group is registered in 600.461, the project will be graded following the 600.461 scheme.

Code and data submission
Please **comment** your code and structure it so that it can be started from a Matlab file named **main.m** (use the matlab cell mode to structure the file if needed). Your code should run on the CS undergrad computers. If manual input is required, please insert the relevant comments in the code and print guidance instructions within the Matlab console (check the Matlab function *disp*). Please also include all necessary data (e.g. images, calibration parameters, training data) within your submission.

Technical note
To create a movie from images and vice-versa, you can take a look at softwares such as *ffmpeg, mencoder, virtualdub* (etc) in addition to the limited *videoreader* and *videowriter* functions from Matlab.
The video *traffic.avi* is available as a stack of *jpeg* images (25 fps) at the following link:
`http://www.cs.jhu.edu/~padoy/media/cvproject2011/traffic-images.tgz`

Have fun !

[6] `http://www.latex-project.org/`