Computer Vision Capstone Project

Build a Social Image Description Platform

Description

Build a social image description platform. Users can upload an image that they would like to have described, the image is then surfaced in other users' *streams* and one or more users describe it. Over time, the system learns associations between words used to describe an image and the visual features found in it, and gets better at predicting descriptions on its own.

You can use one of (or a combination of) several different methods to gather features from images, including (but not limited to):

- Visual Bag of Words
- Gist
- SIFT, SURF or other local feature descriptors
- Shape Context

For labeling, you can treat a description as a simple bag-of-words. A more advanced model could parse the description string entered by a user to understand sentence structure and semantics, and later generate descriptions that sound natural.

Setup

If you are building a desktop application/program or a web application:

- Computer Vision Development Setup and Reference (follow Python instructions)
- Clone this library and install it: Lumos
  Play around with the library, create a copy of the sample code provided in the README, and modify it to do something fun. Make sure it runs as expected; you should be able to call it with a video or static image file as input on the command-line, or process the camera stream (default).

If you are building an Android or iOS application, you may need to use OpenCV for Android/iOS. Alternately, you may only need native functionality or other common libraries - it entirely depends on what algorithms (and how much of them) you want to implement.

Tasks

Use the following instructions as a guideline to work on your projects. Make sure you address all the questions/prompts given below in your write-up.

1. Choose a target platform (desktop/web/Android/iOS), and design your overall application. Describe it as if you are building a proof-of-concept product, complete with user interface design (very simple mockups will do), intended scope, features.
2. Explain clearly the computer vision/machine learning component within your application, including the inputs, desired outputs and some algorithms that you are considering.
3. Build the user-facing of the system first, including a means to upload an image, show unlabeled images in a user's stream/homepage, and store descriptions that are entered.
   Add screenshots of the system to your report explain usage scenarios and flows.
4. Prepare a dataset for training and testing, so that you can repeatedly (and reliably) measure the performance of your system. You can use the system built so far to generate this dataset (perhaps with some help from your friends?!).
   Include statistics about your dataset in the report (no. of image, train-test split, etc.).
5. Now implement and test your image-description matching algorithm.
   What computer vision and/or machine learning techniques did you use? What metric are you using to compare the predicted descriptions with the human-labeled ones?
6. Complete and improve your application by implementing any additional features.
   What does your final system do, and how different is it from the system you initially conceived? How well does it perform on seen vs. unseen examples?

Include in your write-up any other thoughts and reflections you may have had during this process. It also important to note any special dependencies and instructions to run your code.

Learning Resources

Courses

- Object Recognition and Scene Understanding, Antonio Torralba (MIT)
- Introduction to Computer Vision, Aaron Bobick (Georgia Tech and Udacity)
- Introduction to Computer Vision, James Hayes (Brown)

Books

- Computer Vision: Algorithms and Applications, Richard Szeliski
- Computer Vision: Models, Learning, and Inference, Simon Prince