


Project 3: Real-time Object 2-D Recognition

Due Tuesday by 8:59pm **Points** 30

This project is about 2D object recognition. The goal is to have the computer identify a specified set of objects placed on a white surface in a translation, scale, and rotation invariant manner from a camera looking straight down. The computer should be able to recognize single objects placed in the image and identify the object an output image. If provided a video sequence, it should be able to do this in real time.

Setup

For development, you can use this [development image set](https://northeastern.instructure.com/courses/103147/files/12258333/download?download_frd=1)  (https://northeastern.instructure.com/courses/103147/files/12258333/download?download_frd=1). Once you have your system working, you'll need to create your own workspace for real-time recognition. If you have a webcam, the easiest thing to do is set it up on some kind of stand facing down onto a flat surface covered with white paper. If you have only a built-in laptop camera, find a section of white wall in front of which you can hold the objects to be identified. Alternatively, you may be able to stream your phone's camera video to your laptop. If you can get a setup where with a downward facing camera on a white surface, that is the easiest thing to do.

Your system needs to be able to differentiate at least 10 objects of your choice. Pick objects that are differentiable by their 2D shape and are mostly a uniform dark color, similar to the dev image set. The goal of this project is a real-time system, but if that is not feasible given your setup, you can have your system take in a directory of images and process them instead. Make sure it is clear in your report which type of system you build.

If you are working on your own, you must write the code from scratch for at least one of the first four tasks--thresholding, morphological filtering, connected components analysis, or moment calculations. Otherwise, you may use OpenCV functions. If you are working in pairs, you must write the code for two of the first four tasks from scratch.

Tasks

1. Threshold the input video

Using the video framework from the first project (if you wish), start building your OR system by implementing a thresholding algorithm of some type that separates an object from the background. Give your system the ability to display the thresholded video (remember, you can create multiple output windows using OpenCV). Test it on the complete set of objects to be recognized to make sure

this step is working well. In general, the objects you use should be darker than the background, and the whole area should be well-lit.

You may want to pre-process the image before thresholding. For example, blurring the image a little can make the regions more uniform. You could also consider making strongly colored pixels (pixels with a high saturation value) be darker, moving them further away from the white background (which is unsaturated).

Required images: Include 2-3 examples of the thresholded objects in your report

2. Clean up the binary image

Clean up your thresholded image using some type of morphological filtering. Look at your images and decide whether they are displaying noise, holes, or other issues. Pick a strategy using morphological filtering (growing/shrinking) to try to solve the issue(s). Explain your reasoning in your report.

Required images: Include 2-3 examples of cleaned up images in your report. Use the same images as displayed for task 1.

3. Segment the image into regions

The next step is to run connected components analysis on the thresholded and cleaned image to get regions. Give your system the ability to display the regions it finds. A good extension is to enable recognition of multiple objects simultaneously (but this is not required). Your system should ignore any regions that are too small, and it can limit the recognition to the largest N regions.

OpenCV has a connected components function; if you don't write your own, use the one that also returns the stats. If you chose to write your own, both the two-pass connected components algorithm and the region growing methods are good algorithms to know.

Required images: Include 2-3 examples of region maps, ideally with each region shown in a different color. Use the same images as for the prior tasks.

4. Compute features for each major region

Write a function that computes a set of features for a specified region given a region map and a region ID. Start by calculating the axis of least central moment and the oriented bounding box. Any features you use should be translation, scale, and rotation invariant such as moments around the central axis of rotation. Give your system the ability to display at least one feature in real time on the video output. Then you can easily test whether the feature is translation, scale, and rotation invariant by moving around the object and watching the feature value. Start with just 2-3 features and add more later. There is an OpenCV function that will compute a set of moments. If you use it, explain what moments it is calculating in your report.

Required images: Include 2-3 examples of regions showing the axis of least central moment and the oriented bounding box. use the same images as for the prior tasks.

5. Collect training data

Enable your system to collect feature vectors from objects, attach labels, and store them in an object DB (e.g. a file). In other words, your system needs to have a training mode that enables you to collect the feature vectors of known objects and store them for later use in classifying unknown objects. You may want to implement this as a response to a key press: when the user types an N, for example, have the system prompt the user for a name/label and then store the feature vector for the current object along with its label into a file. This could also be done from labeled still images of the object, such as those from a training set.

Explain in your report how your training systems works.

6. Classify new images

Enable your system to classify a new feature vector using the known objects database and a scaled Euclidean distance metric $[(x_1 - x_2) / \text{stdev}_x]$. Feel free to experiment with other distance metrics. Label the unknown object according to the closest matching feature vector in the object DB (nearest-neighbor recognition). Have your system indicate the label of the object on the output video stream. An extension is to detect when an unknown object (something not in the object DB) is in the video stream or provided as a single image.

Required images: include the results from at least one image of each object in your report with the assigned label clearly shown.

7. Implement a different classifier

Implement a different classifier system of your choice. For example, implement K-Nearest Neighbor matching with $K > 1$. Note, KNN matching requires multiple training examples for each object.

8. Evaluate the performance of your system

Evaluate the performance of your system on at least 2 different images of each object. Build a confusion matrix of the results showing true labels versus classified labels and include that in your report. It will save you a lot of time if you print out the confusion matrix in CSV format so you can copy and paste a table in to your report.

9. Capture a demo of your system working

Take a video of your system running and classifying objects.

Extensions

- Write a better GUI to show your system in action and to manage the items to your DB

- Add more than the required ten objects to the DB so your system can recognize more objects.
 - Enable your system to learn new objects automatically by first detecting whether an object is known or unknown, and then collecting statistics for it if the object is unknown. Demonstrate how you can quickly develop a DB using this feature.
 - Experiment with more classifiers and/or distance metrics for comparing feature vectors.
 - Explore some of the object recognition tools in OpenCV.
-

Report

When you are done with your project, write a short report on the [Khoury wiki](https://wiki.khoury.northeastern.edu) (<https://wiki.khoury.northeastern.edu>) that demonstrates the functionality of each task. You will need to adjust the permissions of your home space and your report wiki pages so that other people can see it (by default, no one but you can view them). Your report should have the following structure. Please **do not** include code in your report.

1. A short description of the overall project in your own words. (200 words or less)
 2. Any required images along with a short description of the meaning of the image.
 3. A description and example images of any extensions.
 4. A short reflection of what you learned.
 5. Acknowledgement of any materials or people you consulted for the assignment.
-

Submission

We will be using [Gradescope](https://www.gradescope.com) (<https://www.gradescope.com>) for project code submission. You should receive an invitation to enroll in Gradescope using your Northeastern email address. You will be submitting your code and readme file to Gradescope and writing your report using the wiki.

When you are ready to submit, upload your code and a readme.txt (or readme.md) text file. The readme file should contain the following information.

- Links/URLs to any videos you created and want to submit as part of your report.
- The URL for your wiki report for this project.
- What operating system and IDE you used to run and compile your code.
- Instructions for running your executables.
- Instructions for testing any extensions you completed.
- Whether received an accommodation for the project or you are using one or more time travel days.

For project 3, submit your .cpp and .h (.hpp) files, and readme.txt (readme.md). Note, if you find any errors or need to update your code, you can resubmit as many times as you wish up until the deadline.

As noted in the syllabus, projects submitted by the deadline can receive full credit for the base project and extensions. (max 30/30). Projects submitted up to a week after the deadline can receive full credit for the base project, but not extensions (max 26/30). You also have eight time travel days you can use during the semester to adjust any deadline, using up to three days on any one assignment (no fractional days). If you want to use your time travel days, email the instructor prior to the deadline for which you plan to use them. If you need to make use of the "stuff happens" clause of the syllabus, contact the instructor as soon as possible to make alternative arrangements.

Receiving grades and feedback

After your project has been graded, you can find your grade and feedback on Gradescope. Pay attention to the feedback, because it will probably help you do better on your next assignment.