PS3 Review Session

Jingwei Ji
Overview

Space carving

Single Object Recognition via SIFT

Histogram of Oriented Gradient (HOG)
Visual hull:
an upper bound estimate
Space carving - overview

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
- Form the initial voxels as a cuboid
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette
Space carving - (a) (b) (c)

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
- Form the initial voxels as a cuboid #TODO
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette #TODO
Space carving - (a) (b) (c)

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
- Form the initial voxels as a cuboid
  - Try not just do for loops, too slow
  - Probably useful functions: np.meshgrid, np.repeat, np.tile
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette
  - Critical step: project 3D voxel into 2D
  - Remember using homogenous coordinates for projection
  - Also, better no for loops...

Visual hull: an upper bound estimate
Space carving - (d)

Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
- Form the initial voxels as a cuboid
  - Improvement: tighter bounded cuboid
  - How: do a coarse carving first (we use num_voxels = 4000)
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette

Final Output
Steps:
- Estimate silhouettes of images (could be based on some heuristics, e.g. color)
  - Issues: the quality of silhouettes
  - The silhouette from each camera is not perfect, but the result is ok. Why?
  - Experiment: use only a few of the silhouettes
- Form the initial voxels as a cuboid
- Iterate over cameras and remove the voxels which project to the dark part of each silhouette
SIFT descriptor

David G. Lowe. "Distinctive image features from scale-invariant keypoints." IJCV 60 (2), 04

- Alternative representation for image regions
- Location and characteristic scale $s$ given by DoG detector

1. Compute gradient at each pixel
2. $N \times N$ spatial bins
3. Compute an histogram $h_i$ of $M$ orientations for each bin $i$
4. Concatenate $h_i$ for $i=1$ to $N^2$ to form a $1 \times MN^2$ vector $H$
5. Gaussian center-weighting
6. Normalize to unit norm

Typically $M = 8$; $N = 4$
$H = 1 \times 128$ descriptor
We’ve implemented SIFT descriptor for you, and your task is using it for object recognition.


Be sure to understand why is there a threshold and how to use it.
RANSAC to refine matching

Basic idea: use RANSAC to fit a homography matrix $H$ between two set of key points. Only keep the inliers for matching, remove the outliers.

Think: how many pairs of correspondence do we need in each iteration?
More about RANSAC

Theoretical properties: see lecture notes / slides, figuring out the relations between

- **N**: number of samples
- **e**: outlier ratio
- **s**: minimum number of data points to fit the model
- **p**: probability guaranteed
Single Object Recognition Via SIFT - (d) (e)

Find quantitative relation between two bounding boxes

Input \((u_1, v_1, s_1, \text{theta}_1; u_2, v_2, s_2, \text{theta}_2; x_1, y_1, w_1, h_1)\), find \((x_2, y_2, w_2, h_2, o_2)\).

Keypoint1  Keypoint2  Bbx1  Bbx2 & Orientation
Histogram of Oriented Gradients (HOG) - Overview

**HoG = Histogram of Oriented Gradients**

*Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05*

- Like SIFT, but...
  - Sampled on a dense, regular grid around the object
  - Gradients are contrast normalized in overlapping blocks
Three levels of abstraction:

1. pixel-wise gradients (angle, magnitude) -> histogram in cell
2. histogram in cell -> HoG feature in the block
3. HoG feature in the block -> total HoG feature in the image

See whiteboard
Histogram of Oriented Gradients (HOG) - Overview

- Compute the gradients of image (angles, magnitudes)
- Divide the image into blocks with overlapping, each image contains many blocks
- Divide blocks into cells, each block contains several cells, and each cell contains several pixels.
- Generate the histogram (of shape (nbins,)) of each cell, the concatenation of histograms of cells in a block serves as the feature vector of this block
- Output feature: \((H\_blocks, W\_blocks, cells\_in\_block ** 2 * nbins)\)
See whiteboard

- Compute the gradients of image (angles, magnitudes)
- Divide the image into blocks with overlapping, each image contains many blocks
- Divide blocks into cells, each block contains several cells, and each cell contains several pixels.
- Generate the histogram (of shape (nbins,)) of each cell, the concatenation of histograms of cells in a block serves as the feature vector of this block
- Output feature: (H_blocks, W_blocks, cells_in_block ** 2 * nbins)
Compute the gradients of image (angles, magnitudes)

The way that the angles and magnitude per pixel are computed as follows:
Given the following pixel grid

P1 P2 P3
P4 P5 P6
P7 P8 P9

We compute the angle on P5 as $\arctan(dy/dx) = \arctan\left(\frac{P2-P8}{P4-P6}\right)$.

The magnitude is simply $\sqrt{(P4-P6)^2 + (P2-P8)^2}$.
- Compute the gradients of image (angles, magnitudes)
- Divide the image into blocks with overlapping, each image contains many blocks
- Divide blocks into cells, each block contains several cells, and each cell contains several pixels.
- **Generate the histogram (of shape (nbins,)) of each cell**, the concatenation of histograms of cells in a block serves as the feature vector of this block
- Output feature: (H_blocks, W_blocks, cells_in_block ** 2 * nbins)
def generate_histogram(angles, magnitudes, nbins = 9):

Binning the angles with magnitudes as weights

Be careful about the edge cases: what if the angle is close to 0 or 180 degrees?
Histogram of Oriented Gradients (HOG) - (c)

- Compute the gradients of image (angles, magnitudes)
- Divide the image into blocks with overlapping, each image contains many blocks
- Divide blocks into cells, each block contains several cells, and each cell contains several pixels.
- Generate the histogram (of shape (nbins,)) of each cell, the concatenation of histograms of cells in a block serves as the feature vector of this block
- Output feature: \((H_{\text{blocks}}, W_{\text{blocks}}, cells_{\text{in}}_{\text{block}} \times 2 \times nbins)\)