Christian Marin
Troy Tharpe
Dean Fortier

EEL 4930 Final Project Submission Detail

Files Included:

1. main.py – main script controlling simulation, typically calls core for complex functionality
2. core.py – custom library, abstracts detail and provides higher level interface for main
3. util.py – custom library, contains accessory functions, typically visualization tools used by core

Steps to run code:

1. Navigate to the 'project' subfolder. This should be Spyder's default behavior on main execution.*
2. Ensure all dependencies are installed (numpy, scipy, sklearn, cv2, matplotlib, numba)
3. Ensure necessary data is found in the correct repository subfolders:
    a. data_train folder requires the provided 'data_train.mat' to be added
    b. data_test folder requires the provided 'data_test.mat' to be added
4. Set the SIMULATION_DOWNSAMPLE flag to true for significant downsampling of the data that produces no valuable results but serves to demonstrate the flow of data through the system end to end.
5. Set the SIMULATION_DOWNSAMPLE flag to false for a full-scale simulation that takes about one hour and at least 32 GB of available RAM to run end-to-end. Memory Errors or crashes during periods of peak memory usage like feature extraction and LDA training may occur otherwise.
6. Run main.py (this will make calls core.py and util.py as necessary).

Summary of algorithm:

1. Load in training data consisting of pixel centers for white cars, red cars, pools, and ponds as well as pixel masks for ponds.
2. Run a 15x15 sliding window across the entire image with a single pixel step, extracting:
    a. RGB, HSV, and grayscale color spaces
    b. Binary labels for edges created via canny edge detection with bilateral filtering
    c. FFT real and imaginary components with non-local means denoising
3. Histogram the values found in each window into 8 bins for each space. This aggregates the 225 pixels in each window with values ranging from 0-255 into bins of 0-31, 32-63, etc. This results in a more generalized, robust, and spatially invariant ultimate feature space.
4. Shuffle both pixel level labels and pixel level features according to the same index, so that both are matched but in random order and trim 99.3% of the excessive background points.
5. Train the LDA classifier.
6. Load in test data consisting of RGB pixel values for a new image.
7. Run feature extractor described in steps 2 and 3 above
8. Predict pixel-wise classes using the pretrained LDA model on the newly extracted features.
9. Perform post processing to aggregate nearby points
    a. Blob detection proved too computationally intensive, given time constraints we resorted to a thresholding method to reduce FP rates and hoped to also remove too-near pixels.
10. Score using F1 scoring method discussed in lecture.

Team Indentation Error Final Project

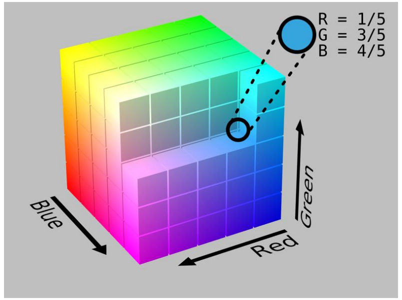
Christian Marin
Troy Tharpe
Dean Fortier

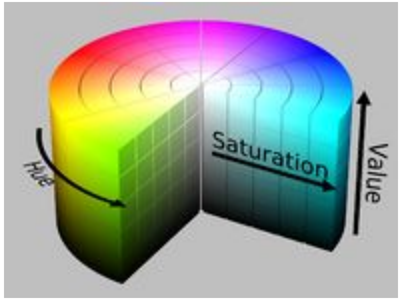
System Outline

- Pre-processing
- Feature Extraction
- Classification
- Post-processing
- Summary

Pre-processing: Color Spaces

- RGB - given, intuitive
- HSV - performed well in preliminary analysis
- GRY - intuitive, emphasis on intensity (technically redundant*)
  - GRY = .2989*R + .5870*G + .1140*B





Pre-processing: Bilateral Filter

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

I^{filtered} is the filtered image;

I is the original input image to be filtered;

x are the coordinates of the current pixel to be filtered;

Ω is the window centered in x;

Effectively smooths texture while preserving edge boundaries better than a standard Gaussian.

Pre-processing: Non-Local μ Denoising

- "Semi-local" filter designed to remove Gaussian noise from images
- Performs a more distant search for "similar" pixels to average in
- Performs particularly well smoothing ponds and fields

Pre-processing: Canny Edges

1. **Intensity Gradient:** convolve with X and its transpose:

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

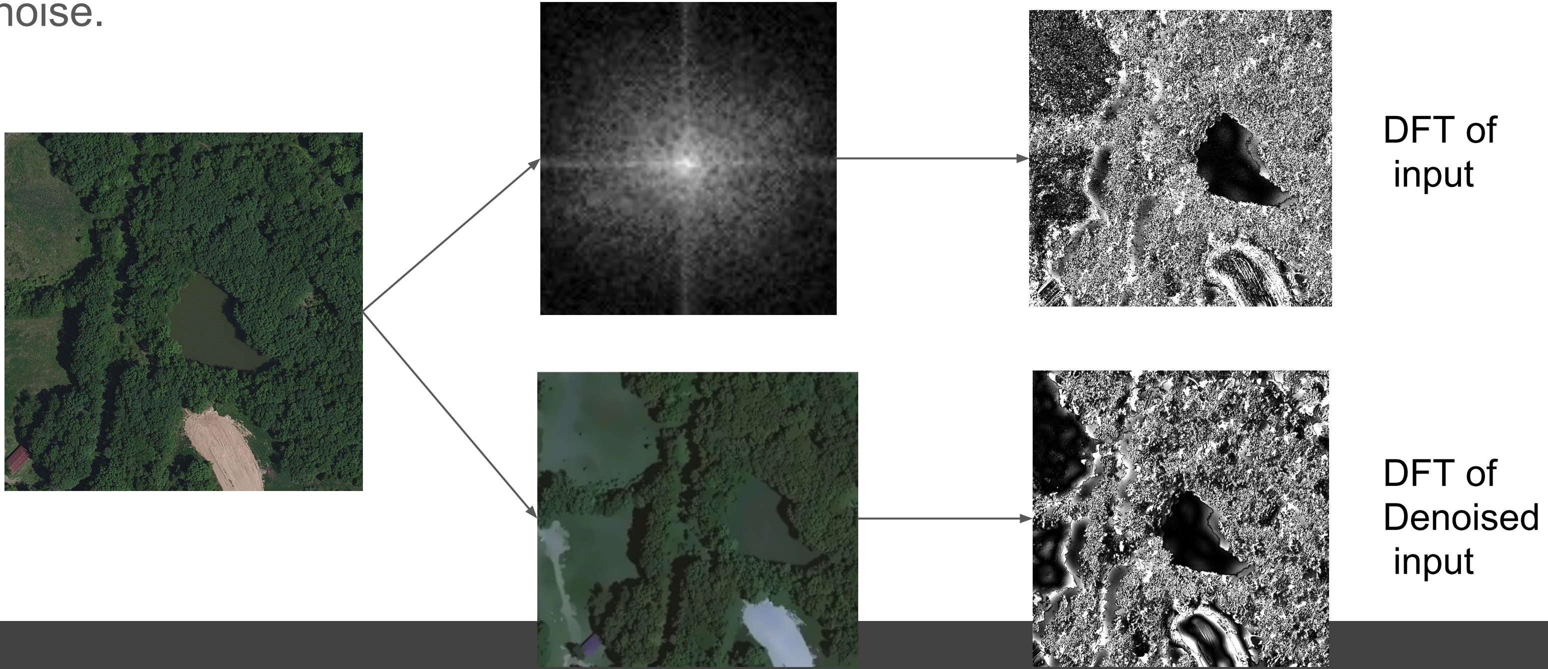
*rounded to nearest axis/main-diagonal

2. **Non-Max Suppression:** compare neighboring intensity gradient magnitudes and force outlying values further toward extremes

3. **Threshold:** cut off high and low values for the most probable and improbable edges

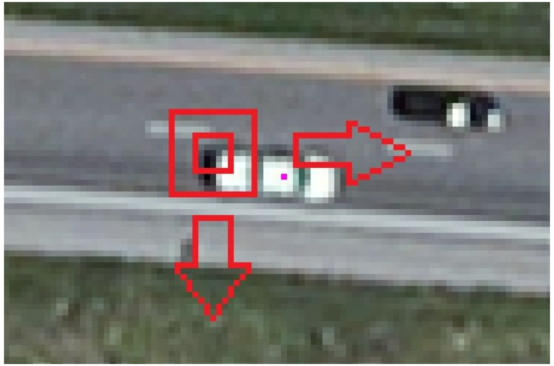
Pre-processing: Fourier Transform

- Transform a grayscale image to 2D frequency domain with a DFT, masking the low frequency components, revert back to image with IDFT with reduced low frequency noise.



Feature Extraction: Histograms

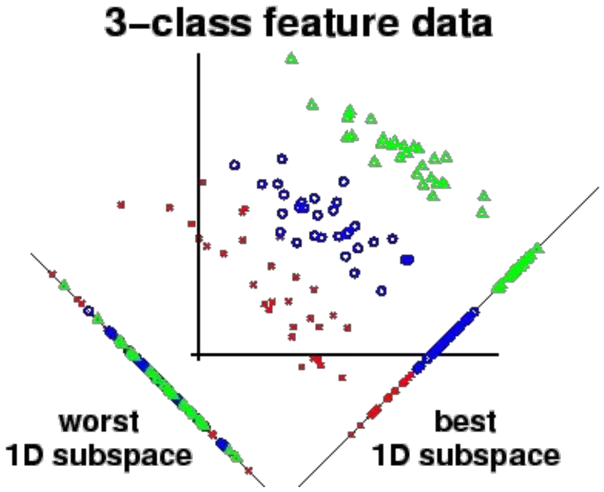
- Sliding window histogram aggregation
- Why not make each pixel a feature?
  - Higher dimensionality
  - Higher variability (aggregation is invariant)
- Append center values to preserve focus
- Loss of precision due to memory
  - data type linear range mapping
  - binning for histograms
- Major computational bottleneck



Classification: LDA

Motivation:

- The problem at hand involves the classification of multiple classes (LDA is generally considered the goto)
- Believed it was safe to assume unimodal Gaussian likelihood within label classes (consistent and characteristic targets)
- Limited training set which would make training models like neural networks more difficult
- Considers within class and between class variance



LDA: Cont.

Implementation:

- Assigned a class to each pixel in image (Red Car, White Car, Pool, Pond, and **Background**).
- Background accounts for more than 99.3% of pixels. In order to avoid over training on background, only 0.7% was included in model fitting
- Accomplished via minimizing within class variance (**Sw**) and maximizing between class separability (**Sb**) according to the objective function:

$$(12)\; J(\mathbf{w}) = \frac{(\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T}{s_1^2 + s_2^2} = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

LDA: Cont.

- The objective function is then minimized by differentiating with respect to weights:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = 0$$

- This results in the weight update equation: $\mathbf{w}^* = \mathbf{S}_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$
    - Where:

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n \in C_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n \in C_2} \mathbf{x}_n$$

    - And: Sw^(-1)= inverted within class variance matrix

Post-processing: Center of Contour

Problem: Needed a mapping of pixel label probabilities to discrete centers

Accomplished with the help of OpenCV

1. Convert image to grayscale
2. Gaussian smooth using a 5x5 kernel
3. Apply a binary threshold (255 = 1, 60 or lower = 0)
4. Calculate closed area contours via border following*
5. Calculate each shape's image moment according to: $M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$
    a. (i, j) = moment order
6. Calculate x and y centroids according to: $C_x = \frac{M_{10}}{M_{00}}$ and $C_y = \frac{M_{01}}{M_{00}}$

Topological Structural Analysis of Digitized Binary Images by Border Following SATOSHI SUZUKI

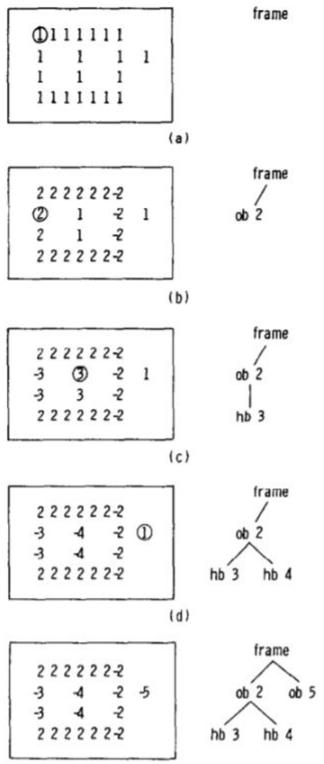
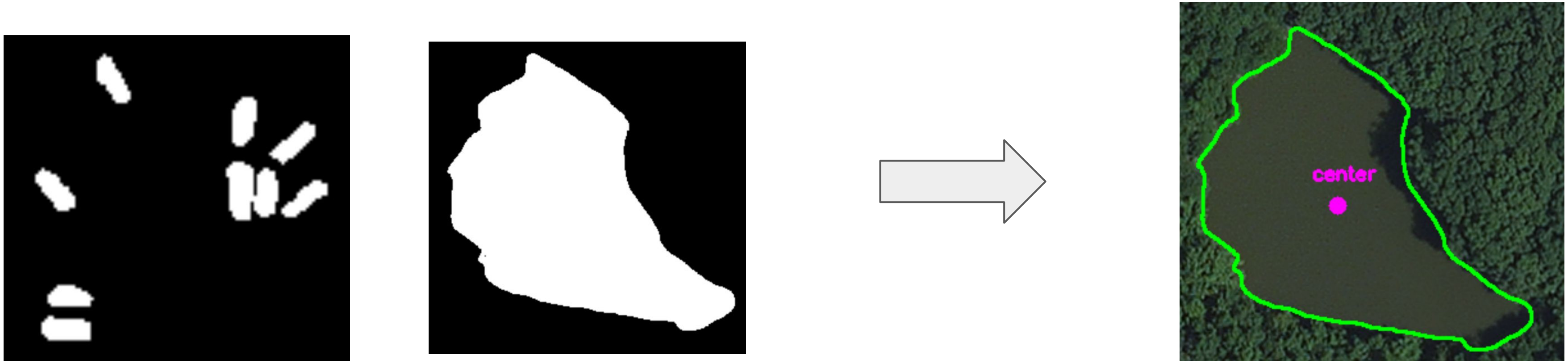


Figure depicting border following algorithm

Center of Contour: Cont.

Results on prelabeled data:

- Green outline depicts recognized contour
- Purple dot depicts centroid of pixel level contour
- Black and white images depict binary label input



Summary

Future considerations include:

- Labeling additional classes to reduce variability of the background class
- Supplementing the edge/gradient features in preprocessing
- Application of other classifiers and ensembling
- Implement area/perimeter features

The End