Vessel Classification using Graph Kernel

Lifan Xu **Marco A. Alvarez** **John Cavazos**

Department of Computer and Information Science
University of Delaware

Outline

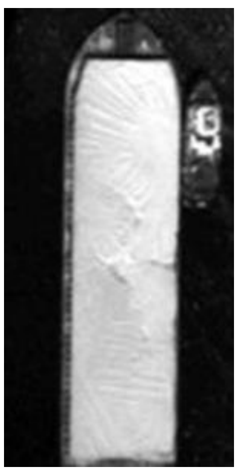
- Introduction
  - Vessel Satellite Images
  - Graph
  - Graph Kernel
- Convert Image to Graph
- Graph Kernels
  - Shortest Path Graph Kernel
  - Unordered Neighboring Graph Kernel
- Results

Outline

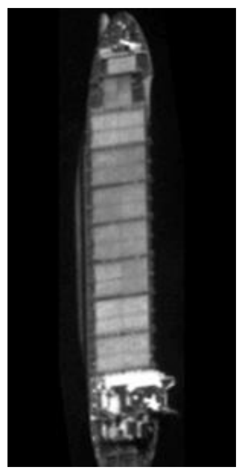
- Introduction
  - Vessel Satellite Images

Vessel Satellite Images

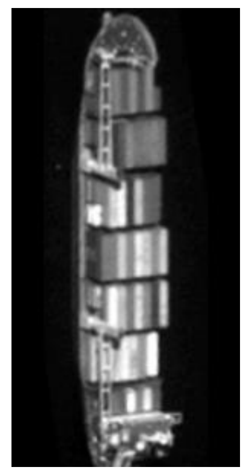
- Four Classes, each class has 200 images



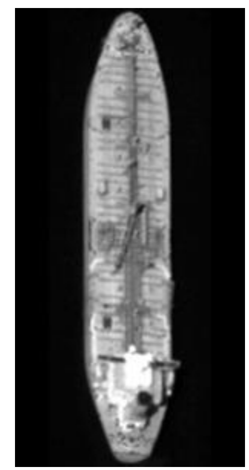
Barge



Cargo



Container



Tanker

Outline

- Introduction
  - Vessel Satellite Images
  - Graph

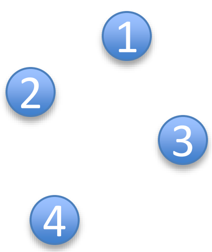
Graph

- A *graph* G is a set of vertices ***V*** and edges ***E***, where ***E*** $\subset$ ***V***^{2}
- A *graph* G may have labels on vertices and/or edges
- The *adjacency matrix* ***A*** of G is defined as

$$[\boldsymbol{A}_{ij}] = \begin{cases} 1 & if\ (v_i, vj) \in \boldsymbol{E} \\ 0 & otherwise \end{cases}$$

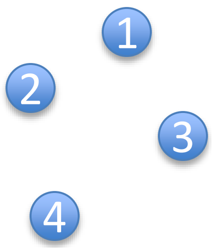
Labelled Undirected Graphs

vertices

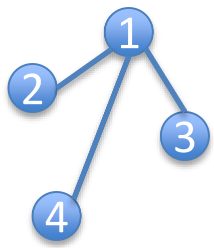


Labelled Undirected Graphs

vertices

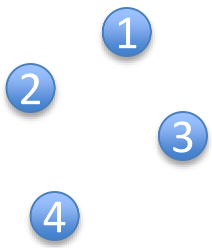


edges

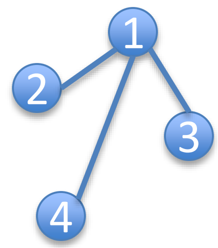


Labelled Undirected Graphs

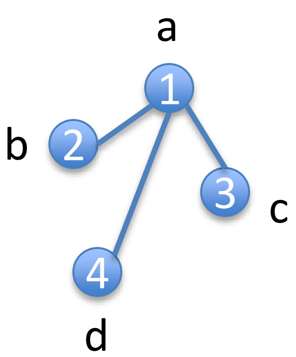
vertices



edges

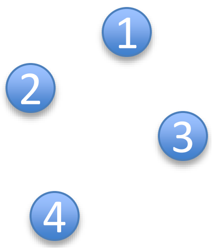


labels

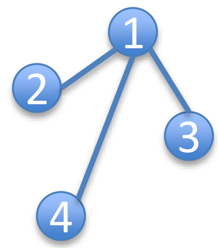


Labelled Undirected Graphs

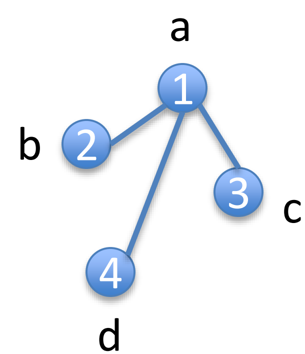
vertices



edges



labels



$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Outline

- Introduction
  - Vessel Satellite Images
  - Graph
  - Graph Kernel

Graph Kernel

- To Calculate the similarities between two graphs in polynomial time
  - Random Walk Kernel
    - Compare all walks in two graphs ***G*** and ***G'***
  - Shortest Path Kernel
    - Compare all pairs shortest paths for ***G*** and ***G'*** via Floyd-Warshall
  - Subtree Kernel
    - Compare subtree-like patterns in two graphs ***G*** and ***G'***
  - Cyclic Pattern Kernel
    - Compare simple cycles in two graphs ***G*** and ***G'***
  - Graphlet Kernel
    - Count subgraphs of limited size *K* in ***G*** and ***G'***

Outline

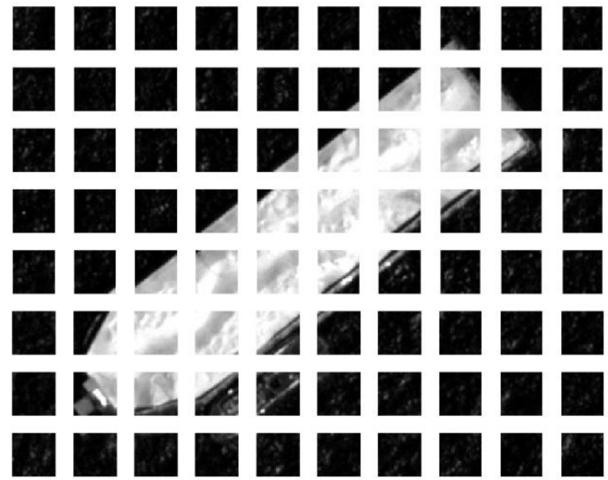
- Introduction
  - Vessel Satellite Images
  - Graph
  - Graph Kernel
- Convert Image to Graph

Convert Image to Graph

- Cut image into tiles
- Feature Extraction on each tile
  - Each tile is represented by a histogram
    - Local Binary Pattern (LBP)
    - Scale Invariant Feature Transform (SIFT)
- Eliminate uninteresting tiles
  - Set a threshold for the mean value of histogram
  - 0.175 in our case
- Connect the interesting tiles to form a graph
  - Each tile is a node
  - Each node connects with its vertical and horizontal neighbors

Cut Image into Tiles

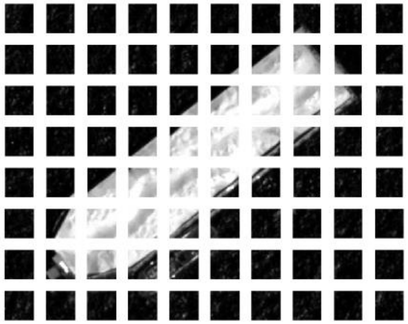
Cut Image into Tiles



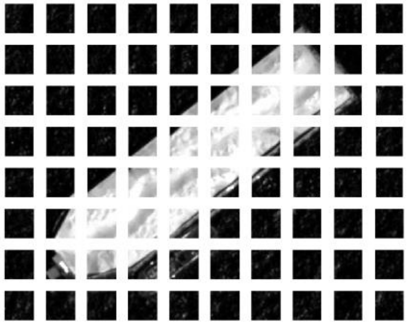
Local Binary Pattern

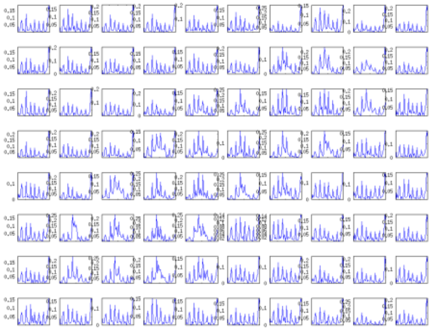
- Each Pixel compares with its eight neighbors
  - Write 1 if the pixel's value greater then neighbor
  - Write 0 otherwise
  - Form a 8-digit binary number(256 patterns)
- Compute histogram of the frequency of each pattern
  - The image is represented by a 256-dimension vector

Feature Extraction using LBP

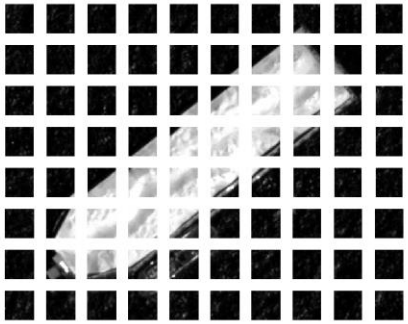


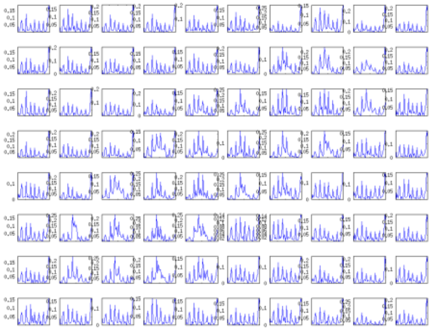
Feature Extraction using LBP

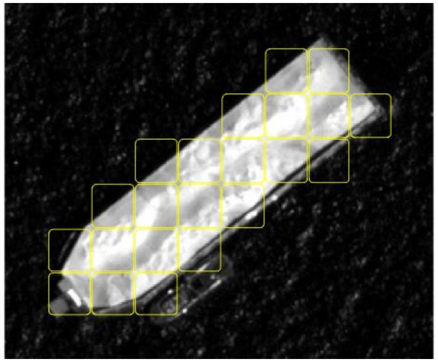




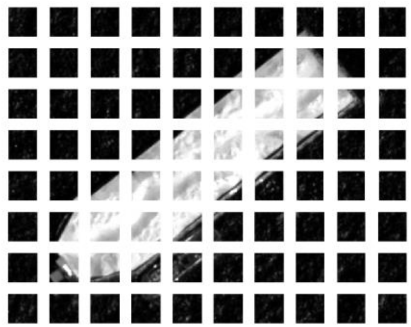
Feature Extraction using LBP

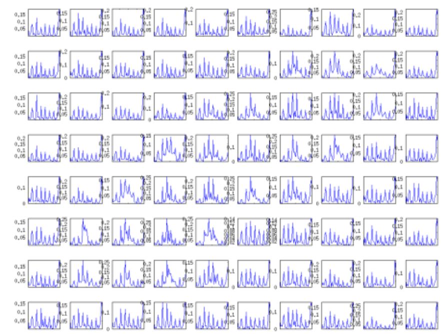


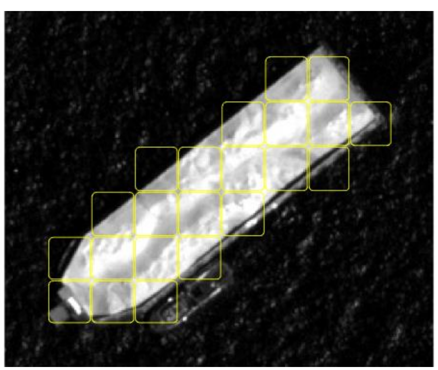


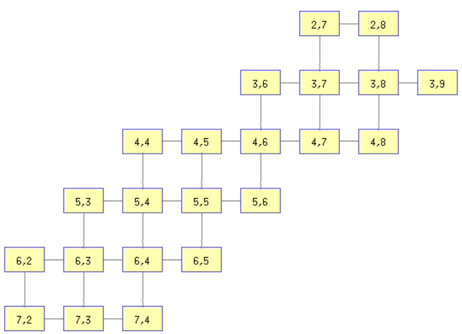


Feature Extraction using LBP





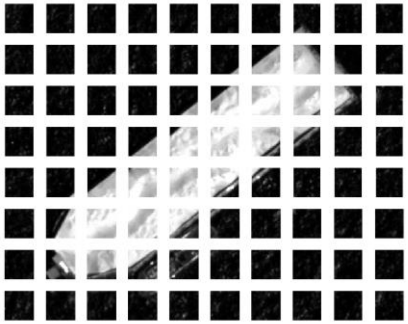




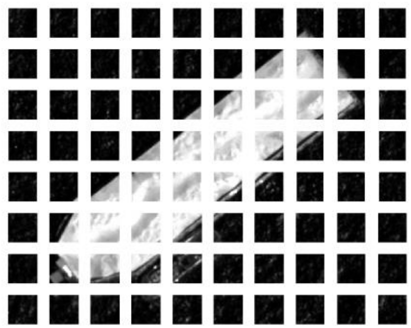
Feature Extraction using SIFT

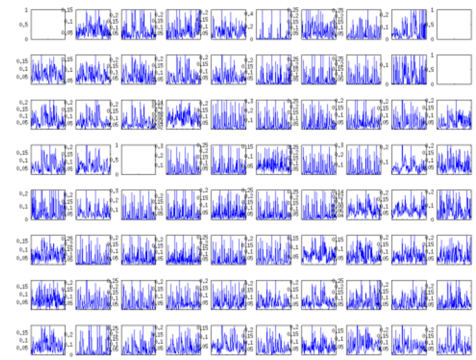
- SIFT Frame
  - (X,Y) coordinates
  - Scale
  - Orientation
- SIFT Descriptor
  - Spatial histogram of image gradients
  - 128-dimension vector
- If a image tile contains multiple SIFT key point
  - Average them to get a mean value 128-dimension vector

Feature Extraction using SIFT

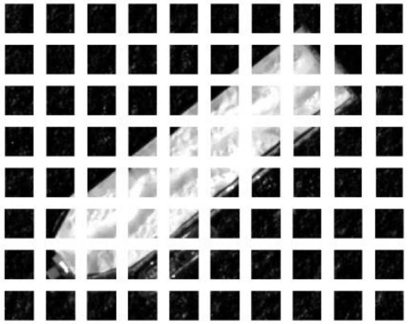


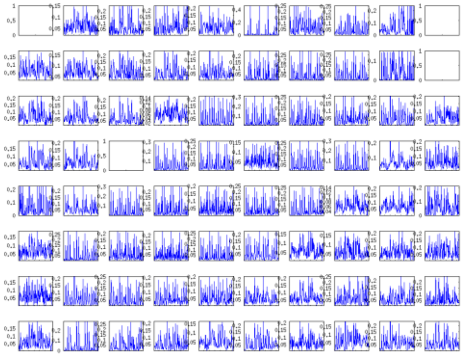
Feature Extraction using SIFT

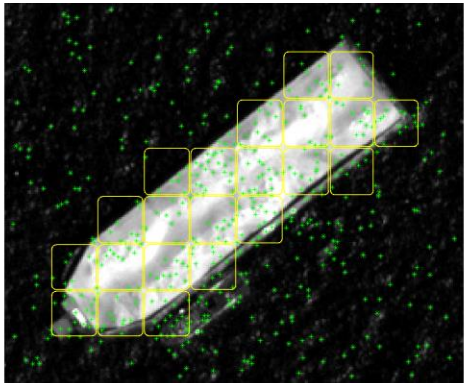




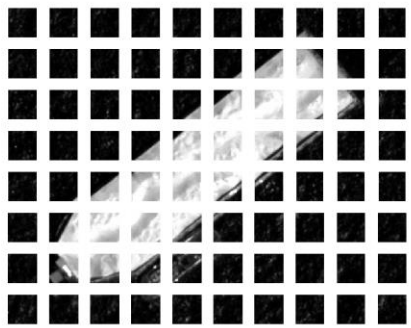
Feature Extraction using SIFT

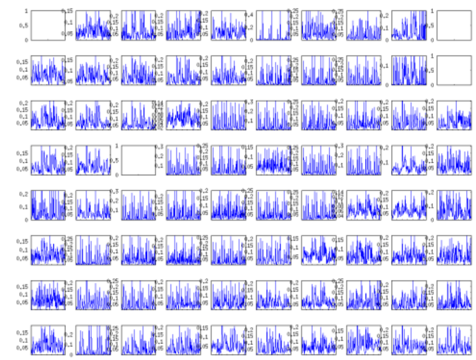


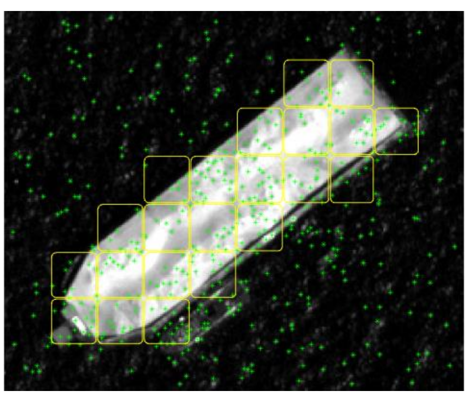


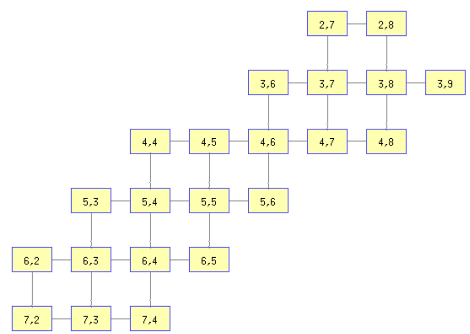


Feature Extraction using SIFT









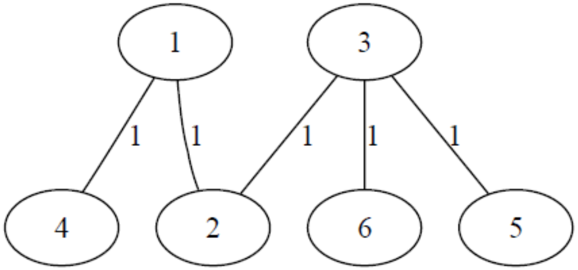
Outline

- Introduction
  - Vessel Satellite Images
  - Graph
  - Graph Kernel
- Convert Image to Graph
- Graph Kernels
  - Shortest Path Graph Kernel

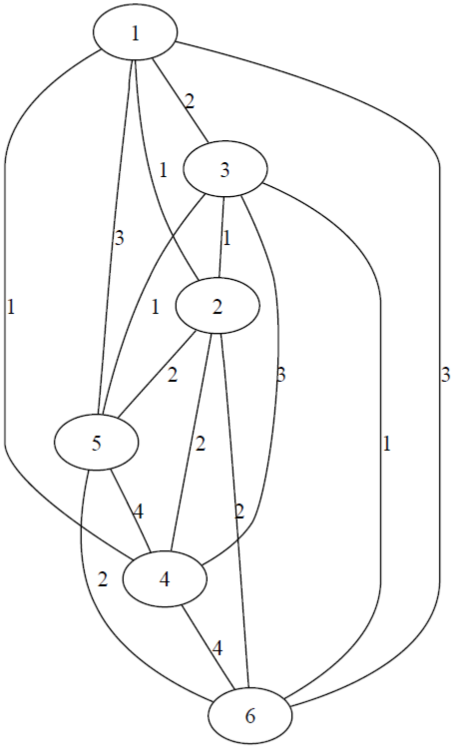
Shortest-Path Graph Kernel

- Convert graph to all pair shortest path graph
  - Floyd-Warshall Algorithm

Floyd-Warshall

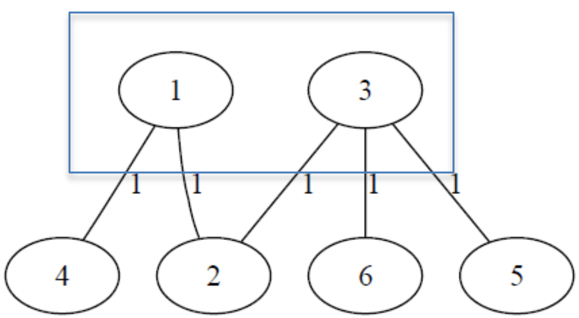


Original Graph

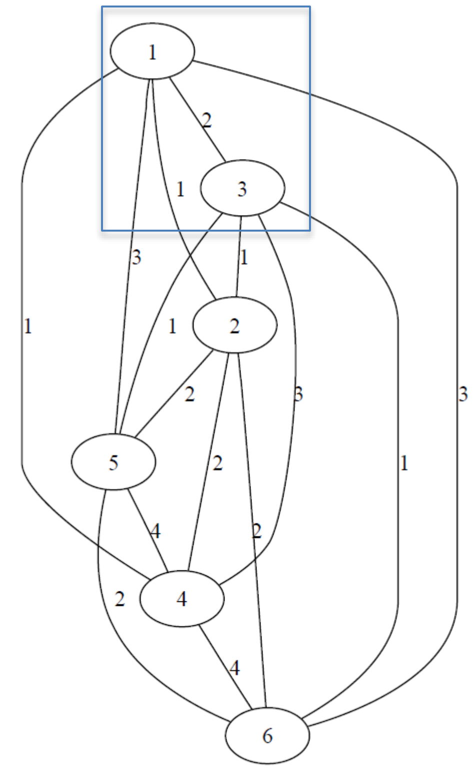


Shortest Path Graph

Floyd-Warshall



Original Graph



Shortest Path Graph

Shortest-Path Graph Kernel

- $K_{sp(G,G')} = \sum_{e \in E} \sum_{e' \in E'} K_{walk}(e, e')$

Shortest-Path Graph Kernel

- $K_{sp(G,G')} = \sum_{e \in E} \sum_{e' \in E'} K_{walk}(e, e')$
- $K_{walk}(e, e') = Knode(u, u') \cdot K_{edge}(e, e') \cdot Knode(v, v')$

Shortest-Path Graph Kernel

- $K_{sp(G,G')} = \sum_{e \in E} \sum_{e' \in E'} K_{walk}(e, e')$
- $K_{walk}(e, e') = Knode(u, u') \cdot K_{edge}(e, e') \cdot Knode(v, v')$
- K_{node} is a valid kernel function for comparing two vertices
  - Gaussian kernel
  - Intersection kernel
- K_{edge} is a valid kernel function for comparing two edges

Outline

- Introduction
  - Vessel Satellite Images
  - Graph
  - Graph Kernel
- Convert Image to Graph
- Graph Kernels
  - Shortest Path Graph Kernel
  - Unordered Neighboring Graph Kernel

Unordered Neighboring Graph Kernel

- Given node v, let N(v) be the set contains all neighbors of node v

$$k_{UNORD}(G, G') = \sum_{v \in V} \sum_{v' \in V'} k_{node}(v, v') * (1 + \sum_{n \in N(v)} \sum_{n' \in N(v')} K_{node}(n, n'))$$

Outline

- Introduction
  - Vessel Satellite Images
  - Graph
  - Graph Kernel
- Convert Image to Graph
- Graph Kernels
  - Shortest Path Graph Kernel
  - Unordered Neighboring Graph Kernel
- Results

Data sets

- BCCT200-original
  - Unchanged images
- BCCT200-rotated
  - images are rotated
- BCCT200-cropped
  - images are rotated and cropped
- BCCT200-resized
  - images are rotated, cropped, aligned and resized

SVM results using LBP feature

Data	SP-Gaussian	SP-Intersect	Unord-Gaussian	Unord-Intersect
BCCT200-original	**79.5**	75.8	76.7	75.1
BCCT200-rotated	81.3	78.2	80.3	75.8
BCCT200-cropped	83.8	80.3	80.9	77.7
BCCT200-resized	88.8	86.9	**89.2**	86.8

SVM results using SIFT feature

Data	SP-Gaussian	SP-Intersect	Unord-Gaussian	Unord-Intersect
BCCT200-original	87.8	84.5	**88.9**	82.7
BCCT200-rotated	87.3	84.2	87.3	83.1
BCCT200-cropped	88.7	85.1	87.7	83.9
BCCT200-resized	91.8	90.3	**92.4**	91.1

Thanks!

Questions?