Road Network Detection and Network Extraction

Submission date: 29-May-2023 02:26AM (UTC-0400)
Submission ID: 2104357762
File name: maj_proj_docs.pdf (1.38M)
Word count: 5872
Character count: 27108

CONTENTS

Abstract

Networks are one of the primary components of many systems. Graph theory provides a precise way to investigate the properties of a network given a graph. Software for analyzing graphs is easily accessible and has been employed to investigate different kinds of networks. While graph acquisition is relatively straightforward in some applications, it can be more complicated for certain networks. For instance, data collections such as road networks rely on images, and extracting graphs from these images often requires specialized solutions. In this article, we present a new tool, Road Network Detection & Network Extraction, which extracts graphs from images of road networks. This tool presents a novel approach by integrating fundamental tools from image processing, computer vision, and graph theory, facilitating the effortless extraction of graphs from images for practitioners. This project serves as an alternative to time-consuming manual graph extraction and specialized tools. We believe that this tool will help expedite the process of collecting large graph datasets from road network image datasets.

Chapter 1: Introduction

The study of complex road networks has become increasingly important for many solutions that require collecting network data from roads. A mathematical method, called Graph Theory, is used for designing graph networks where there are two network components: which are vertices (or nodes), and the pairwise relations between the vertices is called edges. In certain communities, vertices may be also known as sites or nodes, while edges may be known as links or arcs. When a mathematical graph can be extracted with ease from a physical network, the analysis of larger graphs with a large number of vertices can be performed in very less time.

The feasibility of building large databases of various types of networks has made it possible to apply software that incorporates statistical and graph theory methods. As a result, numerous insights have been obtained, leading to a better understanding of the structures of large-scale networks.

However, digitizing road networks remains challenging, and readily-analyzable datasets are often not available. In numerous experimental scenarios, networks are first presented as high-resolution images. Prior to any analysis, it is imperative to acquire a segmented image of the network image, and from there, extract the relevant graph data. This process involves identifying the vertices and edges within the structure, which can quickly become very labor-intensive even for smaller networks, making automated solutions necessary. With the help of the latest advancements in computer vision, the segmented image is processed to compute a skeleton. This skeleton is then utilized to extract the graph that represents the original network.

Chapter 2: Implementation Requirements

In image processing, Python is a highly favored programming language due to its widespread usage. It boasts a diverse range of libraries and tools that simplify the task of working with images and executing various image processing functions. The following are some of the widely used libraries and tools in Python for image processing, as utilized in this project:

1. **NumPy**: NumPy is a Python library utilized for numerical computing in image processing. It facilitates the use of multidimensional arrays, which are advantageous for storing and manipulating images. Typically, NumPy is employed in conjunction with other image processing libraries to perform an array of image processing functions.

2. **OpenCV**: With an open-source computer vision library, OpenCV provides an array of functions for image and video processing. It features a Python interface that enables the library's use in Python code, with functions for image processing tasks such as image filtering, edge detection, object detection, and image segmentation.

3. **NetworkX**: Python package NetworkX aids in the formation, manipulation, and investigation of complex network structures, dynamics, and functions.

4. **cv-algorithms**: Equipped with standard image processing algorithms, such as thinning algorithms (Guo and Hall thinning algorithm & Zhang Suen thinning algorithm), this library is prepped and ready for use.

For input images the Road network detection algorithm needs "clean" Maps Screenshots without landmarks, buildings, any labels or markers. You can use screenshots of google map from https://mapstyle.withgoogle.com/ with the appropriate settings to remove those road-obstructing objects from the screenshots

Chapter 3: Road Network Detection and Road Network Extraction Algorithm

Road Network Detection & Network extraction Algorithm gives us the data regarding the road network in a Graph Data structure. Graph Theory formalizes a network's structure by modeling the constituents of a network as vertices and the pairwise relations between them as edges.

Following are the two main steps in this algorithm:

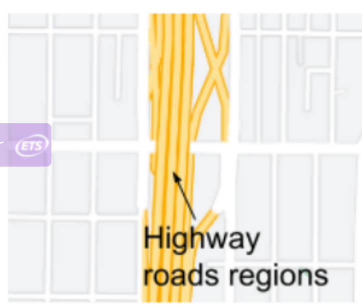
1. Segmenting road network (Network Detection)
2. Obtaining the Graph Data (Network Extraction)

3.1. Network Detection

In this step we perform the segmentation of the road network on the map screenshot image. The conversion of map image to segmented image will look like as follows:

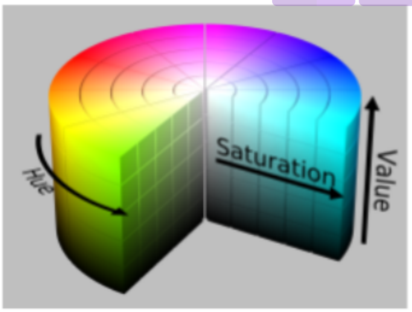
1. Load the map screenshot.

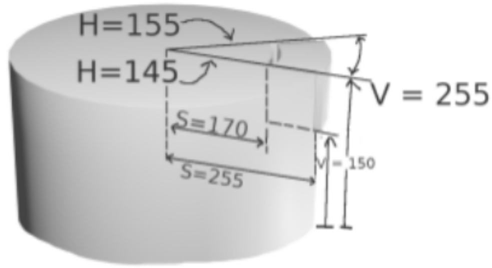
2. We can not use simple thresholding to detect all the roads because highway roads belong to a different color space than regular roads. Also, the highway roads have different hues of yellow. So, use HSV color space to detect the highway roads.



HSV color space is capable of targeting all the colors in the Highway roads given that we provide appropriate upper limit color and lower limit color. Then using the color space get the Highway mask image.

```
lower_color = np.array([145, 170, 150])
upper_color = np.array([155, 255, 255])
mask = cv2.inRange(image, lower_color, upper_color)
```





The following is the output of obtaining the highway road region mask.

3. Using the highway road region mask, we can thereafter replace the highway road colors in the original image with intensities (255, 255, 255) (white color). So we have to make sure our make is clean without any noises

4. Remove Noise from the generated mask image. For this:

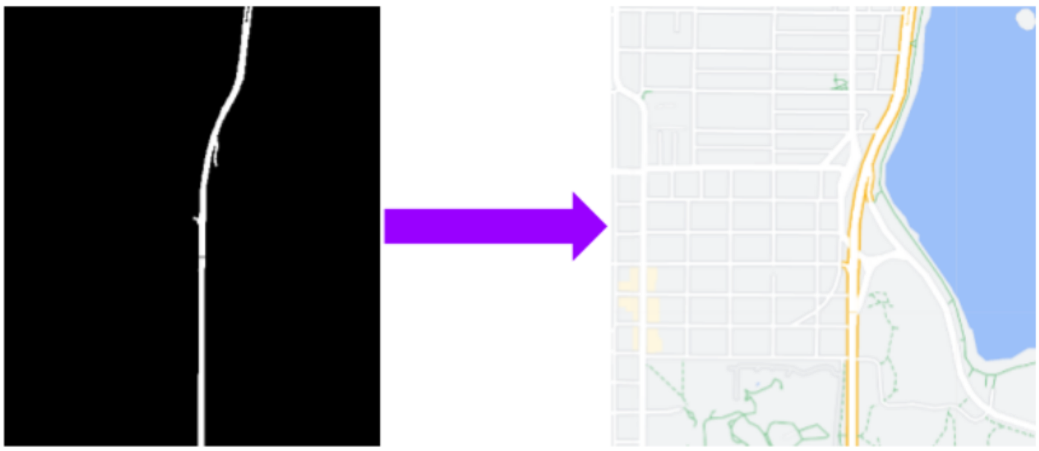
    a. Get the Contour of all the Regions.

    b. Remove regions with small Contour Areas

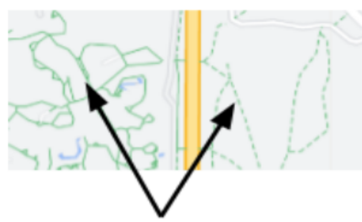
```python
# remove dots which are of small area
contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
for cnt in contours:
    if cv2.contourArea(cnt) < 5:
        cv2.drawContours(mask, [cnt], 0, (0, 0, 0), -1)
```

5. Apply the road region mask on the original image to get the final road network

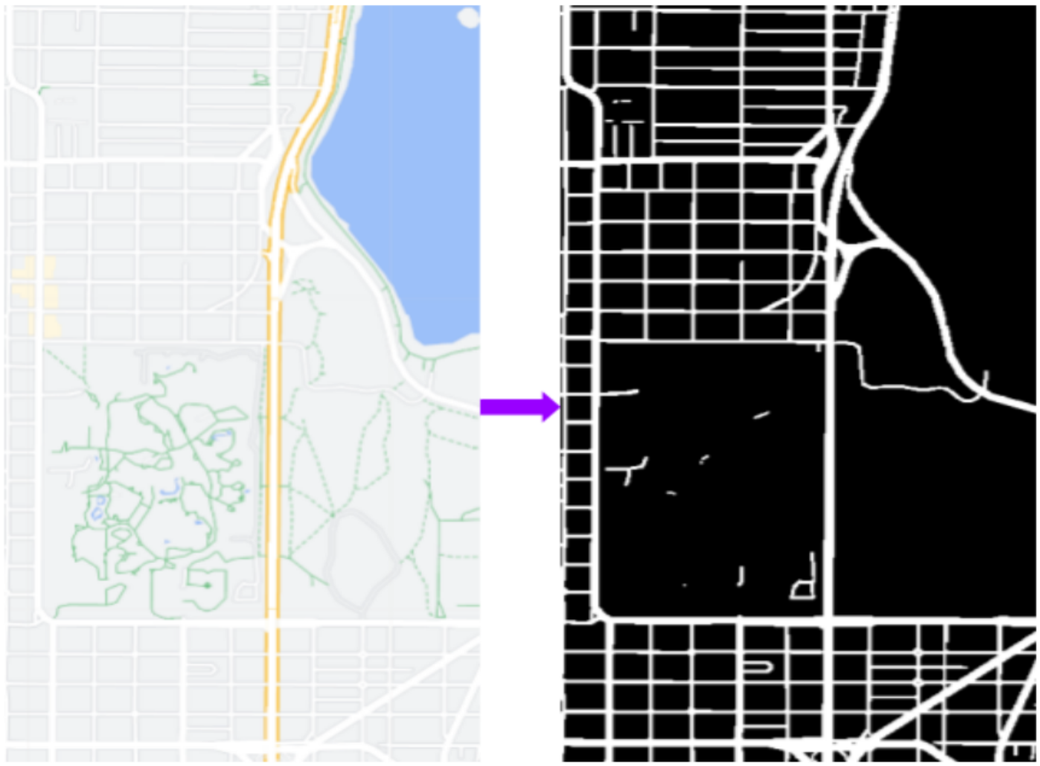
```python
image[mask > 0] = (255, 255, 255)
```



6. By default we are ignoring non-drivable areas. But we can repeat the previous mask generating steps to include these paths.

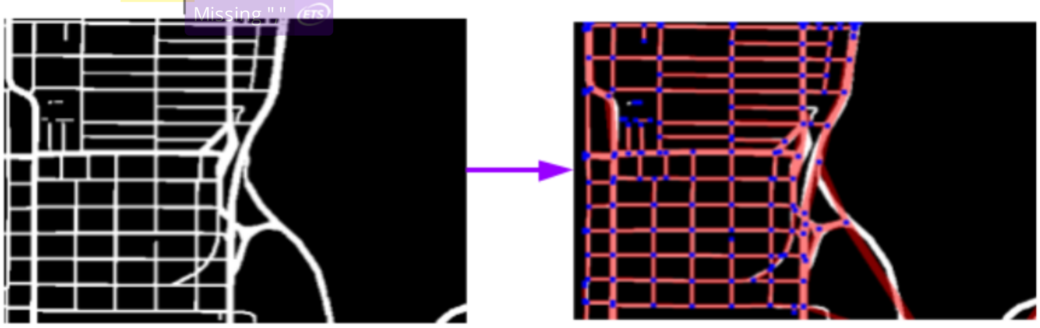


7. Finally, for the Segmentation of our RoI (the Road Network) perform Thresholding at intensity value `250`. All non-RoI areas like water bodies, buildings and other non-road region objects have grayscale intensities less than `250`. Thus they will be given intensity `0` (black) and the RoI areas will be given intensity `255` (white). So, the final conversion will look like as shown below:

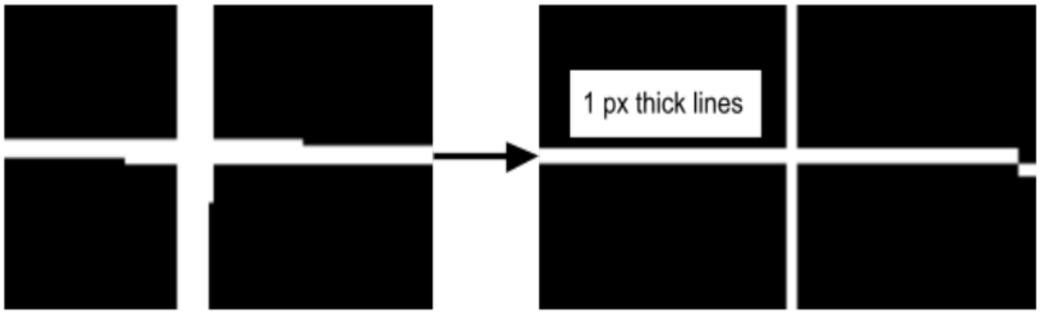


3.2. <u>Network Extraction</u>

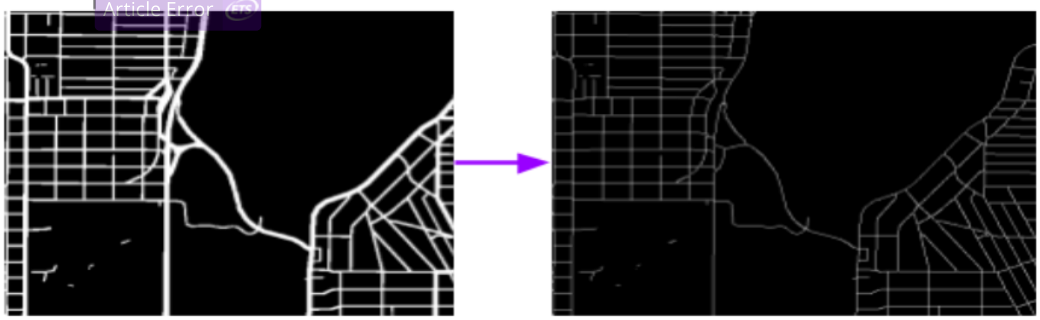
In this step we perform extraction of the road network data in the form of a Graph data structure from the segmented image that was obtained in the Network Detection step. After extraction of the graph data we will plot the graph data on the segmented image/ original map screenshot image which will look as follows:



We can not detect nodes from the image because a lot of nodes will be formed at road intersection points because the road lines are thick (>1px thickness). So, first use the **Zhang Suen thinning algorithm** to perform a thinning operation on the road network segmented image to obtain the skeletonized image.



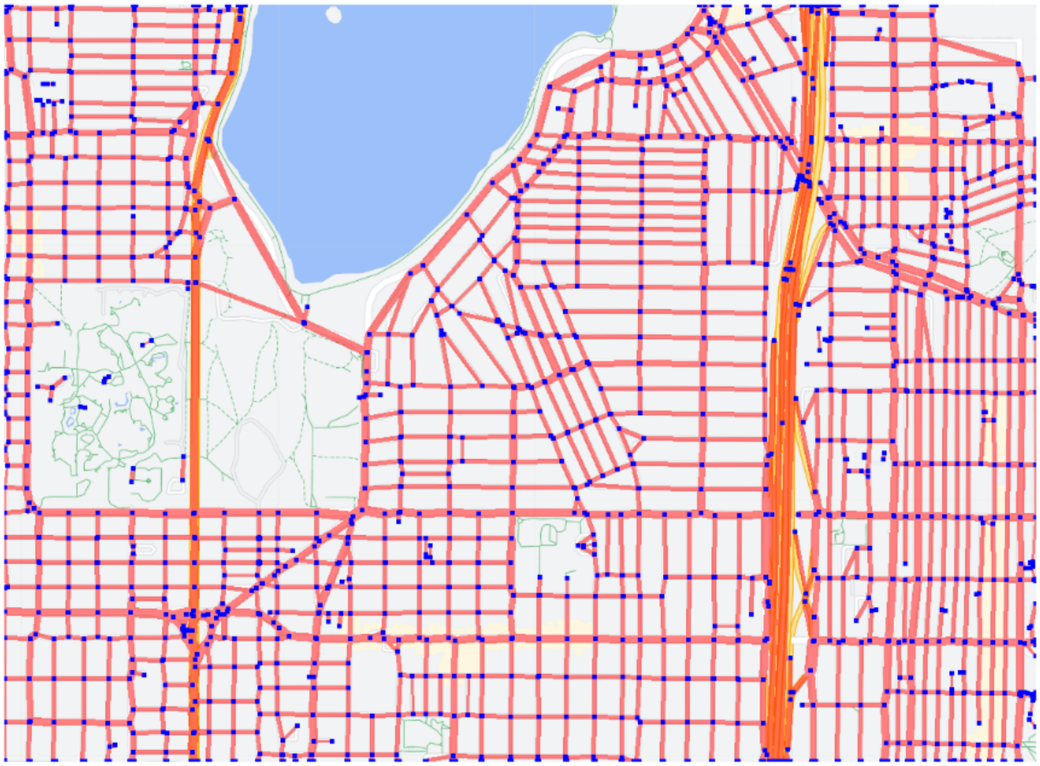
Example of the final skeletonized output image



We perform node detection of the skeletonized image using **Zhang Suen Node detection algorithm** (Discussed separately in chapter 5: Zhang Suen Node Detection algorithm)

For Edge detection, we use a standard graph traversal approach from Graph Theory:- **Breadth First Edge detection** approach (Discussed separately in chapter 6: Breadth First Edge Detection)

Plot the Obtained Graph Data on the original Map screenshot image to visualize the graph data. The conversion finally looks as follows:



Chapter 4: Zhang-Suen Thinning Algorithm

4.1. Introduction

Thinning is a method used in image processing to reduce the width of a binary image while preserving its connectivity and topology. This has a wide range of applications like object recognition, shape analysis, and character recognition.

Thinning transforms objects ~~on a~~ binary image to a set of basic digital lines or arcs that roughly follow the centerline of the objects. This process is not affected by minor bends or irregularities in the image objects. The algorithm recursively deletes simple border points with more than one neighbor. However, it does not delete the endpoints of thin arcs.

The Zhang-Suen thinning algorithm, developed by T. Y. Zhang and S. Y. Suen is a commonly used thinning algorithm. It is an iterative process that thins a binary image by removing pixels from the boundary of foreground objects. The algorithm consists of two passes.

4.2. Algorithm

To understand Zhang-Suen thinning algorithm, let us assume that black pixels represent one, while white pixels represent zero in a rectangular N by M array consisting of ones and zeros. The algorithm targets all black pixels `P1`, which have eight neighboring pixels. The sequence of the neighbors is arranged in a specific order as follows:

P9	P2	P3
P8	P1	P4
P7	P6	P5

It is apparent that the pixels at the edge of the image cannot have eight neighbors. To outline the algorithm, the following definitions are necessary:

`X(P1)` is the number of transitions from white to black (0 to 1) in the sequence `P2, P3, P4, P5, P6, P7, P8, P9, P2`.

`Y(P1)` is the number of black pixel neighbors of `P1`.

Step 1: All the pixels are tested, and those that satisfy the following conditions simultaneously are noted:

1. The pixel is black, and has eight neighbors
2. `2 <= Y(P1) <= 6`
3. `X(P1) = 1`
4. At least any one of `P6`, `P2`, and `P4` is white.
5. At least any one of `P8`, `P4`, and `P6` is white.

After going through the image and collecting all the pixels that satisfy the above conditions 1 to 5, these pixels are set to white.

Step 2: All pixels are again tested, and pixels that satisfy all the following conditions are noted:

1. The pixel is black, and has eight neighbors.
2. `2 <= Y(P1) <= 6`
3. `X(P1) = 1`
4. At least any one of `P8`, `P2`, and `P4` is white.
5. At least any one of `P8`, `P2`, and `P6` is white.

After going through the image and collecting all the pixels that meet the above conditions `1` to `5`, these pixels are set to white. Both passes are iterated until none of the pixels change anymore.

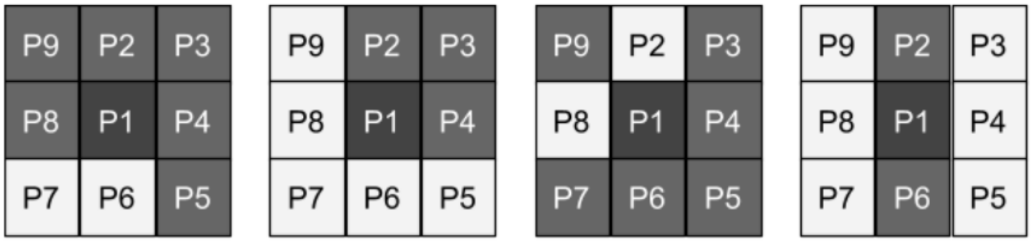
4.3. <u>Algorithm Key Takeaways</u>

To completely understand the working of this algorithm we have to just focus on the five conditions of either step. Consider an image with white background and black foreground.

1. The black pixel `P1` is always at the center of the 3x3 window. The window is scanned from left to right and top to bottom. The pixel `P1` is black and has eight neighbors.
2. The main objective of the 3x3 window is to determine if the pixel `P1` is to be removed (replaced with white) or not by examining the neighboring pixels.
   a. From the third condition, `X(P1) = 1` essentially means that the neighbouring pixels must be black continuously (i.e without any white gaps) when seen in a clockwise or anticlockwise direction and there should be only one continuous white gap.

b. From the second condition `Y(P1) = 2,3,4,5,6` means that the pixel `P1` should have at least 2 and at most 6 black continuous neighbours (i.e without any white gaps) when seen in a clockwise or anticlockwise direction.

c. So one can deduce the good and bad neighbouring pixels pattern in the 3x3 window.



Accepted neighbour pixel pattern in order to remove P1. (Continuous black neighbouring pixels)

Bad neighbour pixel pattern. Pixel P1 won't be removed. (Discontinuous black neighbouring pixels)

3. The *step 1*'s condition 4 and 5 produces 3x3 windows such that the pixel `P1` is removed when the pixel(s) from the bottom (`P6`) and/or right (`P4`) side of the window is white, i.e the thinning is occurring from bottom right side of whole image.
4. Similarly, the *step 2*'s condition 4 and 5 produces 3x3 windows such that the pixel `P1` is removed when the pixel(s) from the top (`P2`) and/or left (`P8`) side of the window is white, i.e the thinning is occurring from top left side of whole image.

Zhang Suen Thinning algorithm is easily available in *cv-algorithms* python package. It can be used as below

```python
import cv_algorithms
skeleton = cv_algorithms.zhang_suen(segmented_image)
```

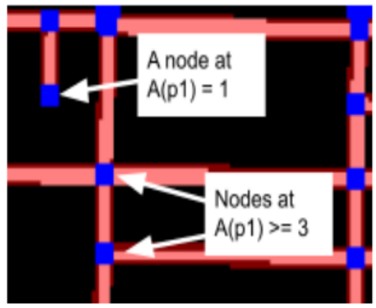
Chapter 5: Zhang-Suen Node Detection Algorithm

5.1. Introduction

Node detection in NEFI1 is based on criterias put forward in "A fast parallel algorithm for thinning digital patterns" by T. Y. Zhang and C. Y. Suen.

Pixel `P1` of the skeleton is categorized as nodes or non-nodes based on the value of a function `A(P1)` depending on the pixel neighborhood of `P1`.

The pixel neighborhood of pixel `p1` can be of three types



1. `A(P1) == 1`: The pixel `p1` sits at the end of a skeleton line, thus a node of degree 1 has been found.
2. `A(P1) >= 3`: The pixel `P1` belongs to a branching point of a skeleton line, thus a node of degree `>= 3` has been found.
3. `A(p1) == 2`: The pixel `P1` sits in the middle of a skeleton line but not at a branching point, thus a node of degree `2` has been found. Such nodes are ignored and not introduced to the graph.

5.2. Algorithm

The algorithm is implemented in the `zhang_suen_node_detection` function below. The function takes a skeleton image as input and returns a graph data structure with the detected nodes. The algorithm is implemented as follows:

```
def zhang_suen_node_detection(skel):
    graph = nx.Graph()
    w, h = skel.shape
    item = skel.item
    for x in range(1, w - 1):
      for y in range(1, h - 1):
        if item(x, y) != 0 and check_pixel_neighborhood(x, y, skel):
           graph.add_node((x, y))
    return graph
```

1. We iterate over all pixels of the skeleton image and check if the pixel is a node or not. If the pixel `P1` is a node, we add it to the graph.
2. The pixel `P1` is called a node if the pixel neighborhood of the pixel is either `1` or is more than `2`. The pixel neighborhood is checked by the function `check_pixel_neighborhood`. The function returns a boolean value depending on the number of neighbours around the pixel. If the number of neighbours is either `1` or more than `2`, the function returns `True`, else `False`.
    a. The `check_pixel_neighborhood` function takes the pixel x and y coordinates and the skeleton image as input.

```python
def check_pixel_neighborhood(x, y, skel):
    accept_pixel_as_node = False
    item = skel.item
    p2 = item(x - 1, y) / 255
    p3 = item(x - 1, y + 1) / 255
    p4 = item(x, y + 1) / 255
    p5 = item(x + 1, y + 1) / 255
    p6 = item(x + 1, y) / 255
    p7 = item(x + 1, y - 1) / 255
    p8 = item(x, y - 1) / 255
    p9 = item(x - 1, y - 1) / 255

    nbs_count = (p2 == 0 and p3 == 1) + (p3 == 0 and p4 == 1) + \
        (p4 == 0 and p5 == 1) + (p5 == 0 and p6 == 1) + \
        (p6 == 0 and p7 == 1) + (p7 == 0 and p8 == 1) + \
        (p8 == 0 and p9 == 1) + (p9 == 0 and p2 == 1)

    if (nbs_count >= 3) or (nbs_count == 1):
        accept_pixel_as_node = True

    return accept_pixel_as_node
```

    b. First for all the `8` neighbors of the pixel `p1`, their intensity values are calculated and stored in variables `P2, P3, P4, P5, P6, P7, P8, P9` in normalized form. The normalized form is `1` if the pixel intensity value is `255`, else `0`.

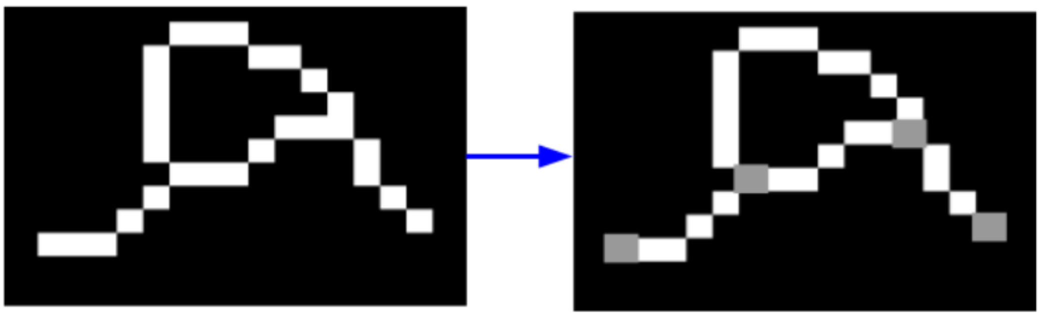
P9	P2	P3
P8	P1	P4
P7	P6	P5

c. The number of neighbors are counted by counting the number of transitions from `0` to `1` or `1` to `0` in the clockwise/anticlockwise direction. The number of transitions is stored in the variable nbs_count.

3. If the function `check_pixel_neighborhood` returns `True`, the pixel is added to the graph data structure as a node.
4. Finally after iterating over all the pixels of the skeleton image, the graph data structure is returned from the `zhang_suen_node_detection` function.

For example, after performing Zhang Suen Node detection in the following skeleton image, we get a graph data structure with nodes as given below:

```
Nodes = [(8, 16), (10, 10), (12, 19), (13, 5)]
```



Chapter 6: Breadth First Edge Detection

The Breadth First Edge Detection algorithm is a method to detect edges in a skeletonized image. It is based on the breadth first search algorithm.

6.1. Algorithm

The algorithm is implemented in `breadth_first_edge_detection` function. The function takes a skeletonized image and a graph as input. The graph contains the nodes of the skeletonized image. The function returns the graph with the edges added to it.

The edges are stored as a tuple of two nodes. The nodes are stored as a tuple of two integers. The integers represent the x and y coordinates of the node in the image.

```
def breadth_first_edge_detection(skel, graph):
    # Prepare queues to perform BFS for edge detection on them
    label_node = dict()
    queues = []
    label = 1
    label_length = dict()
    for x, y in graph.nodes():
       for a, b in neighbors(x, y, skel):
          label_node[label] = (x, y)
          label_length[label] = get_px_length(x, y, a, b)
          queues.append((label, (x, y), [(a, b)]))
          label += 1
    # Breadth first search for edge detection
    edges = set()
    edge_trace = np.zeros(skel.shape, np.uint32)
    edge_value = edge_trace.item
    edge_set_value = edge_trace.itemset
    label_histogram = defaultdict(int)

  while queues
    new_queues = []
    for label, (px, py), nbs in queues:
      for (ix, iy) in nbs:
        value = edge_value(ix, iy)
        if value == 0:
          edge_set_value((ix, iy), label)
          label_histogram[label] += 1
          label_length[label] += get_px_length(px, py, ix, iy)
          new_queues.append((label,(ix, iy),neighbors(ix, iy, skel)))
        elif value != label:
          edges.add((min(label, value), max(label, value)))
    queues = new_queues
```

```python
  # Add edges to graph if they are between different nodes
  for l1, l2 in edges:
    u, v = label_node[l1], label_node[l2]
    if u == v:
      continue
    graph.add_edge(u, v,
         pixels=label_histogram[l1] + label_histogram[l2],
        length=label_length[l1] + label_length[l2])

    # Return graph with edges
  return graph
```

The breadth first search algorithm is implemented as follows:

1. For each node in the graph, first the neighbors of the node are computed. The neighbors are the pixels that are adjacent to the node. The neighbors are computed using the function neighbors. The function takes the `x` and `y` coordinates of a node as input and returns the neighbors of the node. The neighbors are computed as follows

```python
def neighbors(x, y, skel):
    item = skel.item
    width, height = skel.shape
    for dy in [-1, 0, 1]:
        for dx in [-1, 0, 1]:
            if (dx != 0 or dy != 0) and \
                0 <= x + dx < width and \
                0 <= y + dy < height and \
                item(x + dx, y + dy) != 0
              yield x + dx, y + dy
```

2. For each neighbor of the node, a label is assigned to them. The label is a unique integer. The label is stored in a dictionary `label_node`. The key of the dictionary is the unique label and the value is the node. The label is assigned to the neighbor as follows

```python
label_node[label] = (x, y)
```

3. The neighbors of the node are stored in a queue. The queue is a list of tuples. Each tuple contains the label of the node, the node and the neighbors of the node as a list.
The queue will be used for the breadth first search algorithm to detect the edges.

So, the queue is initialized as follows:

```
queues = []
label = 1
for x, y in graph.nodes():
    for a, b in neighbors(x, y, skel):
        queues.append((label, (x, y), [(a, b)]))
        label += 1
```

4. Optionally, the length of the edges can be computed. The length of the edge is stored in a dictionary `label_length`. The key of the dictionary is the unique label and the value is the length of the edge. The length is equal to `1` if the neighbor pixel is adjacent to the node pixel. The length is equal to `1.414214` if the neighbor pixel is diagonal to the node pixel.

   So, The length of the edge is computed as follows:

```
def get_px_length(x, y, a, b):
    return 1.414214 if abs(x - a) == 1 and abs(y - b) == 1 else 1
```

5. The breadth first algorithm for edge detection is implemented as follows

```
# Breadth first edge detection
edges = set()
edge_trace = np.zeros(skel.shape, np.uint32)
edge_value = edge_trace.item
edge_set_value = edge_trace.itemset

while queues:
  new_queues = []
  for label, (px, py), nbs in queues:
    for (ix, iy) in nbs:
      value = edge_value(ix, iy)
      if value == 0:
        edge_set_value((ix, iy), label)
        label_histogram[label] += 1
        label_length[label] += get_px_length(px, py, ix, iy)
        new_queues.append((label, (ix, iy), neighbors(ix, iy, skel)))
      elif value != label:
        edges.add((min(label, value), max(label, value)))
  queues = new_queues
```

The algorithm shown above does the following:

a. Create an `edge_trace` matrix, a matrix of the same size as the skeletonized image. The matrix is initialized with zeros. The matrix is used to store the labels of the nodes. The labels are stored in the matrix at the position of the node in the skeletonized image.

b. Our objective is to perform a breadth first algorithm for edge detection for which we take the queue we created in *step 3* as input. The queue is a list of tuples. Each tuple contains the label of the node, the node coordinates and the neighbors of the node as a list.

c. Iterate over the queue. For each tuple in the queue:

   i. Iterate over the neighbors of the node. For each neighbor:

      1. Verify if the neighbor pixel (that belongs in the skeleton image) is already marked in the `edge_trace` matrix. If the neighbor pixel is not marked, we mark the neighbor pixel with the node's label.

         a. Append the neighbor pixel to the new queue `new_queues`. The neighbor pixel is appended to the queue as a tuple. The tuple contains the label of the node, the neighbor pixel and the neighbors of the neighbor pixel. This `new_queues` will be used to perform the next iteration of the breadth first search at the next level of the graph (Note that, the next level pixels are the neighbors of the current level pixels. We get those neighbour pixels using the function neighbors)

         b. Optionally we can also increment the `label_histogram` with 1 for the label of the node (which is required to calculate the number of pixels on the edge in the skeleton in later steps).

         c. Optionally we can also increment the `label_length` with the length of the edge for the label of the node (which is required to calculate the length of the edge in later steps using the function `get_px_length`).

      2. If the neighbor pixel is already marked in the `edge_trace` matrix, we verify if the neighbor pixel is marked with the same label as the label of the node. If the neighbor pixel is marked with the same label as the label of the node, we do nothing. If the neighbor pixel is not marked with the same label as the label of the node, we add the edge to the graph.

         a. The edge is added to the `edges` set as a tuple of two labels. The labels are sorted in ascending order. This edges set has the possible edges in the skeletonized image which is to be verified later on to add the verified edges to the graph.

   ii. Update the queue with the new queue `new_queues` and repeat the above steps until the queue is empty.

6. Add the edges to the graph. The edges are added to the graph as follows

```
for l1, l2 in edges:
    u, v = label_node[l1], label_node[l2]
    if u == v:
        continue
    graph.add_edge(u, v,
       pixels=label_histogram[l1] + label_histogram[l2],
       length=label_length[l1] + label_length[l2])
    return graph
```

   a. Iterate over the `edges` set. For each edge, we check if the edge that we are going to form is between two completely different nodes (instead of being from the same node to the same node). To do this check we perform the following steps:

      i. To get the coordinates of the nodes in the edge, we use the node's label from the edge tuple. The labels are used to get the coordinates of the nodes from the dictionary `label_node`. If the coordinates of the nodes are the same, we skip the edge because the edge is from the same node to the same node.

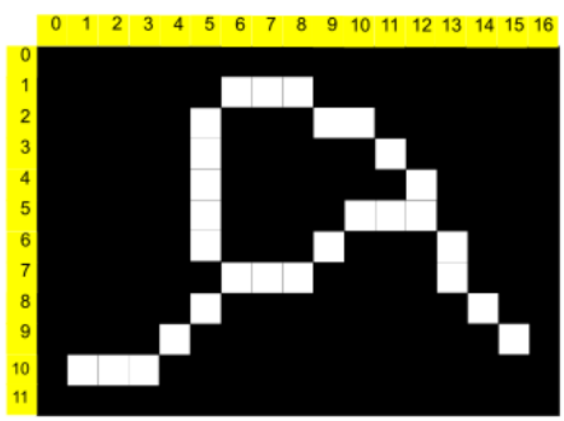
      ii. If the coordinates of the nodes are not the same, we add the edge to the graph. The edge is added to the graph as a tuple of two nodes. The nodes are stored as a tuple of two integers. The integers represent the x and y coordinates of the node in the image.

7. Finally, return the graph which contains the edges of the skeletonized image from the `breadth_first_edge_detection` function.

6.2. Example of algorithm's working

Consider a skeleton image that is as shown.

The pixel coordinates are zero indexed.



Using Zhang-Suen Node detection algorithm we have the pixel coordinates of the nodes of the skeletonized image's graph as shown:

```
>>> print(graph.nodes())
[(12, 5), (6, 7), (15, 9), (1, 10)]
```

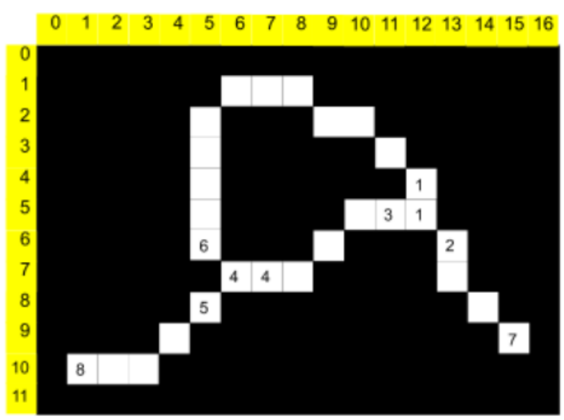
Following the steps of the algorithm:

1. First for each node, we get the neighbors of the node and create the `label_node` dictionary & `queues` queue as previously mentioned in step 2 & step 3.

```
>>> print(label_node)
[ 1: (12, 5),
  2: (12, 5),
  3: (12, 5),
  4: (6, 7),
  5: (6, 7),
  6: (6, 7),
  7: (15, 9),
  8: (1, 10)]
```

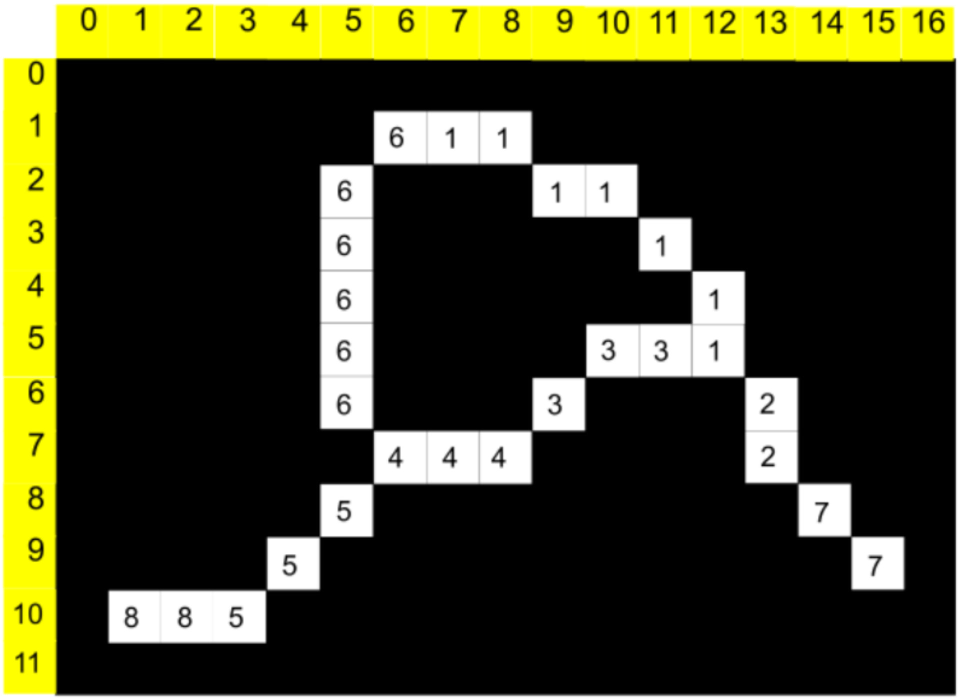
```
>>> print(queues)
[(1, (12, 5), [(12, 4)]),
 (2, (12, 5), [(13, 6)]),
 (3, (12, 5), [(11, 5)]),
 (4, (6, 7), [(7, 7)]),
 (5, (6, 7), [(5, 8)]),
 (6, (6, 7), [(5, 6)]),
 (7, (15, 9), [(14, 8)]),
 (8, (1, 10), [(2, 10)])]
```

2. Now following step 5, we will have a initial `edge_trace` matrix using the first `queues` queue as shown in the image



3. After the subsequent steps new queues were created by performing the algorithm, for which the `edge_trace` matrix got filled up with labels wherever necessary. we finally get the final `edge_trace` matrix with the `edges` set as follows:

   a. Image showing `edge_trace` matrix with integers representing labels. (Consider all black pixels with integer value 0)



   b. `edge_trace` matrix in its 2D matrix form:

```
>>> print(edge_trace)
[[ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 0, 6, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 6, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 3, 3, 1, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 6, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 0, 4, 4, 4, 0, 0, 0, 0, 2, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0],
 [ 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0],
 [ 0, 8, 8, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

c. `edges` set representing possible edges of the graph

```
>>> print(edges)
{(1, 3), (1, 2), (4, 5), (4, 6), (2, 7), (3, 4), (5, 8),
(1, 6)}
```

4. Using the `label_node` dictionary we determined that label pairs `(1, 3), (1, 2), (4, 5), (4, 6)` are not valid edges. So, due to the logic in step 6 they are discarded and not added to the graph.

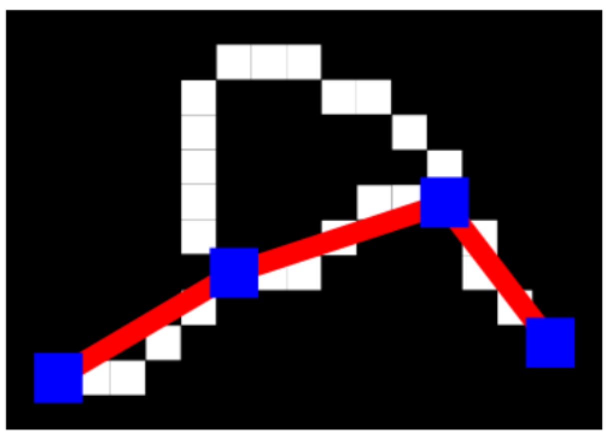
5. The *valid* label pairs are represented by their coordinates as follows:

```
(2, 7) -> ((12, 5), (15, 9))
(3, 4) -> ((12, 5), (6, 7))
(5, 8) -> ((6, 7), (15, 9))
(1, 6) -> ((12, 5), (6, 7))
```

6. So, The final edges of the graph are as follows:

```
>>> print(graph.edges())
[((12, 5), (15, 9)),
 ((12, 5), (6, 7)),
 ((6, 7), (15, 9)),
 ((12, 5), (6, 7))]
```

7. The final graph with edge and nodes can be plotted as follows



Here blue pixels represent the detected node (using Zhang-Suen Node detection algorithm) and red lines represent the detected edges (using Bread First Edge detection algorithm)

6.3. Application of the obtained Graph data

The graph data contains the nodes and edges of the road network. The nodes are the coordinates of the road intersections and the edges are the roads between the nodes. The edges contain the length of the road and the coordinates of the road pixels. The graph data can be used to find the shortest path between two nodes. The shortest path can be found using Dijkstra's algorithm. Dijkstra's algorithm is implemented in the NetworkX library. The shortest path is found using the following code:

```
dijkstra_path = nx.dijkstra_path(graph, (42, 75), (341,
700), weight='length')
```

The above code finds the shortest path between the two nodes with coordinates `(42, 75)` and `(341, 700)`.

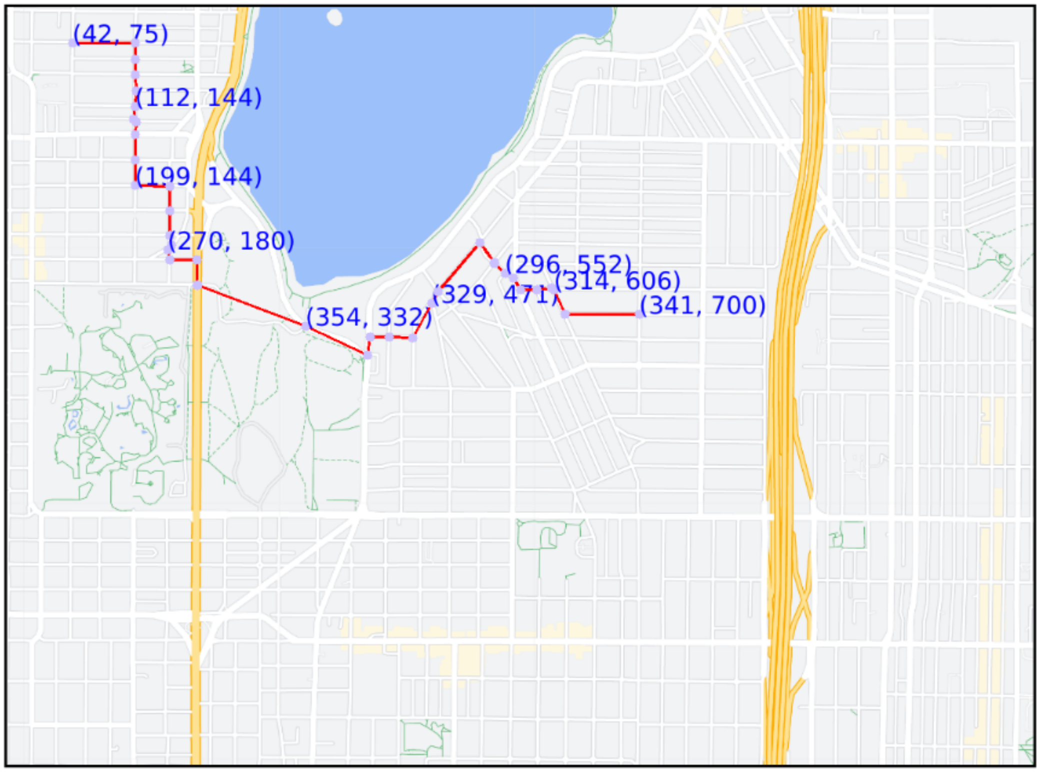
The shortest path is found using the length of the edges. In *Section 6.1*, we saw in step 5 that we were recording the pixel length in the edge's `length` property. Therefore we put `weight='length'` in the `weight` parameter of the `dijkstra_path` function.

The shortest path is drawn on the image using the following code:

```
for i in range(len(dijkstra_path)-1):
    x1, y1 = dijkstra_path[i]
    x2, y2 = dijkstra_path[i+1]
    cv2.line(image_orig, (y1, x1), (y2, x2),
          (0, 0, 255), 2)
    cv2.circle(image_orig, (y1, x1), 5,
          (255, 192, 203), -1)
```

The above code draws the shortest path on the image as a red line. The nodes of the shortest path are drawn as pink circles.

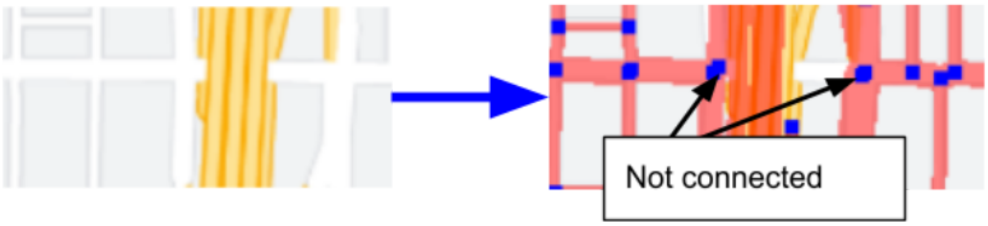
Here the starting point is at `(42, 75)` and the ending point is at `(341, 700)`.



Similarly there can be many applications of the graph data for complex road network analysis. Instead of iterating through millions of pixels to perform road network analysis, we can use the graph data to perform the analysis. The graph data is much smaller in size compared to the image data. Therefore the graph data can be used to perform road network analysis in real-time.

Chapter 7: Current Limitations

1. We can not distinguish between road intersections and flyover roads and due to this some faulty nodes and edges are formed



2. The Input road network image should be a "clean" input roadmap with no road line obstructing objects

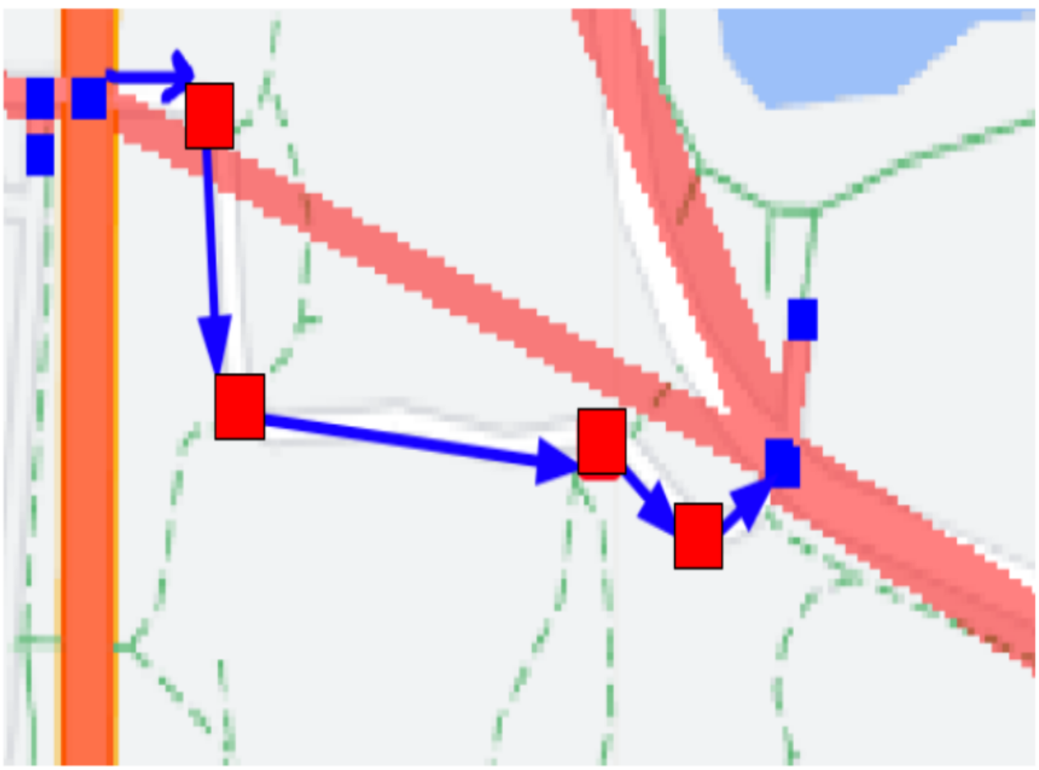
3. The Algorithm only performs well if maps are zoomed in adequately (when all arterial and local roads are visible)

Chapter 8: Future Improvement Work

Currently we are not placing nodes on the curved road regions. We are just placing nodes at the road intersecting and road terminating points. To place nodes on curved roads we have to consider the following possible approach:

Pixel `p1` of the skeleton image is categorized as nodes or non-node based on the value of a function `A(p)` depending on the pixel neighborhood of `p1`.

1. `A(p1) == 1`: The pixel p1 sits at the end of a skeleton line, thus a node of degree 1 has been found.
2. `A(p1) == 2`: The pixel p1 sits in the middle of a skeleton line but not at a branching point, thus a node of degree 2 has been found. Such nodes are ignored and not introduced to the graph.
3. `A(p1) > 2`: The pixel p1 belongs to a branching point of a skeleton line thus a node of degree > 2 has been found.



We can enable node detection for pixels with degree of freedom (df) = 2 on the thinned image to place nodes on curved roads. (Currently only df = 1 & >2 is enabled in the code implementation)

Then, we can utilize the **Ramer–Douglas–Peucker algorithm** to reduce the number of nodes on the curved roads.

Chapter 9: Conclusion

The Road Network Extraction Algorithm is a powerful tool for extracting road networks from images, but its effectiveness is heavily dependent on the quality of the input image. After conducting a thorough analysis of the algorithm's performance, we have come to the conclusion that the segmentation step of the image is the most critical factor in determining the quality of the extracted graph. However, we also found that the errors introduced by the thinning and graph detection steps are minimal when compared to those caused by segmentation.

Therefore, it is essential to ensure that the image being processed is clean and uncluttered for the Road Network Extraction Algorithm to operate at its best. In cases where the input image is more complex, the algorithm can still be used, but the quality of the output may be reduced. Another alternative is to outsource the segmentation step to specialized third-party software to remove the objects that clutter the input image before passing it to the algorithm for graph detection.

References

1. **NEFI 2 Research papers:**
    a. https://www.nature.com/articles/srep15669#Sec5
    b. https://nefi.mpi-inf.mpg.de/guide.php
2. **Zhang-Suen thinning algorithm:** https://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm
3. **Zhang-Suen node detection:** https://www.rsc.org/suppdata/c7/bm/c7bm00227k/c7bm00227k1.pdf?cv=1
4. **Color detection using HSV color model:** https://stackoverflow.com/questions/50210304/change-the-colors-within-certain-range-to-another-color-using-opencv
5. **Ramer–Douglas–Peucker algorithm:** https://en.wikipedia.org/wiki/Ramer–Douglas–Peucker_algorithm

Road Network Detection and Network Extraction

ORIGINALITY REPORT

18%	16%	7%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.rsc.org Internet Source	7%
2	github.com Internet Source	3%
3	www.igi-global.com Internet Source	2%
4	dx.doi.org Internet Source	1%
5	Submitted to MCAST Student Paper	1%
6	www.ijcaonline.org Internet Source	1%
7	www.coursehero.com Internet Source	<1%
8	Wilhelm Burger, Mark J. Burge. "Digital Image Processing", Springer Science and Business Media LLC, 2022 Publication	<1%

Submitted to Curtin University of Technology

9	Student Paper	<1 %
10	Submitted to Harrisburg University of Science and Technology Student Paper	<1 %
11	Rajarshi Das, Roshan Kumar Ojha, Dipam Tamuli, Swapnil Bhattacharjee, Niranjan Jyoti Borah. "Chapter 5 Hand Gesture-Based Recognition System for Human–Computer Interaction", Springer Science and Business Media LLC, 2023 Publication	<1 %
12	Submitted to Stockhom University & The Royal Institute of Technology Student Paper	<1 %
13	Submitted to VIT University Student Paper	<1 %
14	core.ac.uk Internet Source	<1 %
15	www.geeksforgeeks.org Internet Source	<1 %
16	es.slideshare.net Internet Source	<1 %
17	www.ncbi.nlm.nih.gov Internet Source	<1 %

18	Robin Strand. "Shape Representation wtih Maximal Path-Points for Path-Based Distance", 2007 5th International Symposium on Image and Signal Processing and Analysis, 2007 Publication	<1 %
19	ebin.pub Internet Source	<1 %
20	rosap.ntl.bts.gov Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

Road Network Detection and Network Extraction

PAGE 1

Article Error You may need to use an article before this word. Consider using the article **the**.

Article Error You may need to use an article before this word.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Article Error You may need to use an article before this word.

PAGE 2

PAGE 3

Article Error You may need to remove this article.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

S/V This subject and verb may not agree. Proofread the sentence to make sure the subject agrees with the verb.

S/V This subject and verb may not agree. Proofread the sentence to make sure the subject agrees with the verb.

PAGE 4

S/V This subject and verb may not agree. Proofread the sentence to make sure the subject agrees with the verb.

Missing "," You may need to place a comma after this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Missing "," You may need to place a comma after this word.

PAGE 5

Article Error You may need to remove this article.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word. Consider using the article **the**.

Article Error You may need to use an article before this word.

Article Error You may need to use an article before this word.

Article Error You may need to remove this article.

Article Error You may need to remove this article.

PAGE 6

Article Error You may need to remove this article.

S/V This subject and verb may not agree. Proofread the sentence to make sure the subject agrees with the verb.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Sentence Cap. Remember to capitalize the first word of each sentence.

Missing Punct. You are missing a punctuation mark at the end of this sentence.

PAGE 7

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word. Consider using the article **the**.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to remove this article.

Article Error You may need to use an article before this word.

Article Error You may need to use an article before this word.

Missing "," You may need to place a comma after this word.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Article Error You may need to use an article before this word.

Article Error You may need to use an article before this word.

Proofread This part of the sentence contains a grammatical error or misspelled word that makes your meaning unclear.

P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.

Run-on This sentence may be a run-on sentence. Proofread it to see if it contains too many independent clauses or contains independent clauses that have been combined without conjunctions or punctuation. Look at the "Writer's Handbook" for advice about correcting run-on sentences.

Article Error You may need to remove this article.

PAGE 12

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Prep. You may be using the wrong preposition.

Proofread This part of the sentence contains a grammatical error or misspelled word that makes your meaning unclear.

PAGE 13

Proofread This part of the sentence contains a grammatical error or misspelled word that makes your meaning unclear.

Wrong Form You may have used the wrong form of this word.

Article Error You may need to use an article before this word. Consider using the article **the**.

Wrong Form You may have used the wrong form of this word.

Article Error You may need to use an article before this word. Consider using the article **the**.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.

Missing "," You may need to place a comma after this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sentence Cap. Remember to capitalize the first word of each sentence.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

PAGE 15

P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Missing "," You may need to place a comma after this word.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

PAGE 16

P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.

PAGE 17

Missing "," You may need to place a comma after this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Verb This verb may be incorrect. Proofread the sentence to make sure you have used the correct form of the verb.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word. Consider using the article **the**.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

PAGE 18

Article Error You may need to use an article before this word. Consider using the article **the**.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Sentence Cap. Remember to capitalize the first word of each sentence.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Prep. You may be using the wrong preposition.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Prep. You may be using the wrong preposition.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sentence Cap. Remember to capitalize the first word of each sentence.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word. Consider using the article **the**.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

PAGE 20

Article Error You may need to remove this article.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Frag. This sentence may be a fragment or may have incorrect punctuation. Proofread the sentence to be sure that it has correct punctuation and that it has an independent clause with a complete subject and predicate.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Prep. You may be using the wrong preposition.

Verb This verb may be incorrect. Proofread the sentence to make sure you have used the correct form of the verb.

Prep. You may be using the wrong preposition.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Article Error You may need to use an article before this word.

PAGE 21

Article Error You may need to use an article before this word. Consider using the article **the**.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

PAGE 22

Article Error You may need to use an article before this word.

Confused You have used **)]), (** in this sentence. You may need to use **an** instead.

PAGE 23

Possessive You may need to use an apostrophe to show possession.

Article Error You may need to use an article before this word.

P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.

Article Error You may need to use an article before this word.

PAGE 24

P/V You have used the passive voice in this sentence. Depending upon what you wish to emphasize in the sentence, you may want to revise it using the active voice.

Article Error You may need to use an article before this word. Consider using the article **the**.

Prep. You may be using the wrong preposition.

PAGE 25

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Article Error You may need to use an article before this word.

Proper Noun If this word is a proper noun, you need to capitalize it.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Proper Noun If this word is a proper noun, you need to capitalize it.

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Proper Noun If this word is a proper noun, you need to capitalize it.

PAGE 26

Article Error You may need to remove this article.

Article Error You may need to remove this article.

PAGE 27

PAGE 28

Wrong Article You may have used the wrong article or pronoun. Proofread the sentence to make sure that the article or pronoun agrees with the word it describes.

Missing "," You may need to place a comma after this word.

PAGE 29

Article Error You may need to use an article before this word.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

Sp. This word is misspelled. Use a dictionary or spellchecker when you proofread your work.

PAGE 30

Verb This verb may be incorrect. Proofread the sentence to make sure you have used the correct form of the verb.

PAGE 31