Curricula Enrichment delivered through the Application of Location-based Services to Intelligent Transport Systems

Co-funded by the Erasmus+ Programme of the European Union

Teaching & Learning Manual
Step-by-Step Guide:

Spatial Analysis Techniques Image Semantic Segmentation, Space Syntax, and Isovist Analysis

Prepared by
Department of Town & Country Planning
University of Moratuwa, Sri Lanka

2024

Authors

1. Amila Jayasinghe
2. Harini Sawandi

Reviewed and edited by

1. Samith Madusanka
2. Chethika Abenayake

Reviewed by

1. XXXXX (University Name)
2. XXXXX (University Name)

This manual was produced by the Department of Town & Country Planning, with invaluable support of the Erasmus+ Capacity Building in Higher Education (CBHE) project 'Curricula Enrichment for Sri Lankan Universities delivered through the application of Location-Based Services to Intelligent Transport Systems' (LBS2ITS https://lbs2its.net/)

Project Number: 618657-EPP-1-2020-1-AT-EPPKA2-CBHE-JP
Programme: Erasmus+
Key Action: Cooperation for innovation and the exchange of good practices
Action Type: Capacity Building in Higher Education
Co-funding: Erasmus+ Programme of the European Union

Description

This comprehensive manual is an indispensable educational asset tailored for both undergraduate and postgraduate degree programs, presenting an intricate, step-by-step elucidation of diverse spatial analysis techniques pivotal in spatial planning and transport planning. Seamlessly bridging the chasm between theoretical comprehension and pragmatic implementation, this guide is meticulously curated to cater to the requisites of students, educators, and professionals alike.

Immersed within the manual are exhaustive directives on three pivotal spatial analysis techniques: Image Semantic Segmentation, Space Syntax Analysis, and Isovist Analysis. Each technique is meticulously elucidated, offering guidance on data collection, analysis, and interpretation within varied urban contexts.

Whether you're grappling with navigating intricate urban landscapes or seeking to refine your comprehension of spatial dynamics, this manual serves as an indispensable compass, equipping you with the requisite knowledge and proficiency to navigate the complexities of urban planning effectively.

.

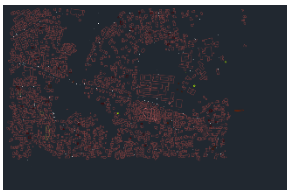
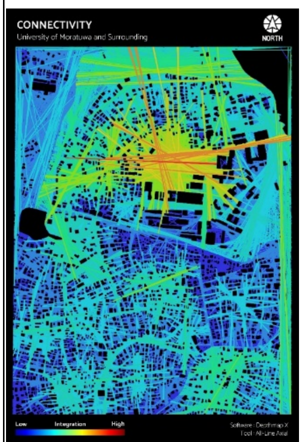
TABLE OF CONTENTS

TABLE OF FIGURE

LIST OF TABLES

SUMMARY OF THE MANUAL

The manual provides in-depth studies using Semantic Image Segmentation, Isovist analysis, and Space Syntax approaches, providing significant knowledge into the spatial layouts, visible patterns, and human behavior in architectural and urban situations.

Analysis Name	Purpose of the Manual	Software Required	Input Data	Output
Image Semantic Segmentation	In the study of Street View factors, ground-level features of **the built environment are analyzed** through image segmentation**.**	**Google Colab** Cloud-based Jupyter Notebook environment for Python. Web Browser: Google Chrome, Firefox, Safari, etc.	**Image datasets for the Segmentation**  The dataset needs to contain RGB images in common image formats such as JPEG and PNG. Each image has a high resolution (e.g., 1024x1024 pixels) to capture fine details and variations in the scene. The dataset can consist of a diverse set of scenes captured from various environments, including urban, rural, and natural landscapes.	**Segmented Image**  Classifying and assigning labels to individual pixels in an image.
Space Syntax Analysis	For urban **spatial arrangement analysis,** Space Syntax is employed as the methodology.	**AutoCAD** **Depth map** Alternatively, ArcGIS with Space Syntax Extension & QGIS with Space Syntax Plugin	**AutoCAD Layer of the case study**  To ensure accurate analysis, proper AutoCAD layers containing boundary elements and obstacles are required as input data.	

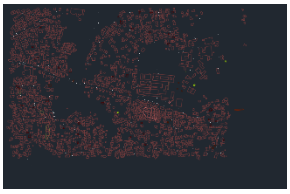
Analysis Name	Purpose of the Manual	Software Required	Input Data	Output
IsoVist Analysis	Isovist Analysis examines **how the geometry and layout of urban environments** influence visibility, movement, and perception.	**AutoCAD** **Isovist App** Alternatively, Depth Map Isovist 3d - Grasshopper Isovist Algorithms	**AutoCAD Layer of the case study**  To ensure accurate analysis, proper AutoCAD layers containing boundary elements and obstacles are required as input data.	 

Table 1 - Summary of the Analysis

1. SEMANTIC IMAGE SEGMENTATION

1.1 What is Semantic Image Segmentation?

Semantic segmentation refers to the process of classifying and assigning labels to individual pixels in an image. Semantic segmentation offers a comprehensive understanding of the scene at the pixel level.

Original Image

Figure 1 -Semantic Segmentation

Segmented Image

Image Segmentation, Semantic Segmentation, Instance Segmentation, and Panoptic Segmentation

https://www.youtube.com/watch?v=5QUmIXBb0MY

Some of the notable areas where image segmentation is used:

- **Face Recognition**

The facial recognition technology present in the iPhone and advanced security systems uses image segmentation to identify faces.

- **Number Plate Identification**

Number plate identification is commonly employed by traffic lights and cameras to impose penalties for investigations. Number plate recognition technology enables a traffic system to identify a vehicle and retrieve its ownership-related data. The system employs image segmentation to isolate a license plate and its associated data from other objects within its field of view.

- **Image-Based Search**

Google and other search engines utilize image segmentation algorithms to detect the items inside an image and then match these findings with comparable photos to provide search results.

- **Medical Imaging**

Image segmentation is employed in the medical field to precisely find and identify cancer cells, quantify tissue volumes, conduct virtual surgery simulations, and facilitate intra-surgery navigation.

- **Remote Sensing and Satellite Image Analysis:**

Segmentation is a technique used in digital image processing to enable satellite images to conduct thorough analysis for activities such as land cover classification or environmental monitoring. Segmentation algorithms play a significant role in obtaining information from a wide range of intricate datasets.

- **Augmented Reality (AR) and Virtual Reality (VR):**

Image segmentation techniques play a crucial role in augmented reality (AR) and virtual reality (VR) applications. It is employed here to separate and engage with scenic components. Utilizing segmentation methods enhances the experience in dynamic virtual settings.

Apart from these applications, image segmentation has uses in manufacturing, agriculture, security, and many other sectors. As our computer vision technologies become more advanced, the use of image segmentation techniques will increase accordingly.

1.2 Process and the Stages of the Semantic Image Segmentation

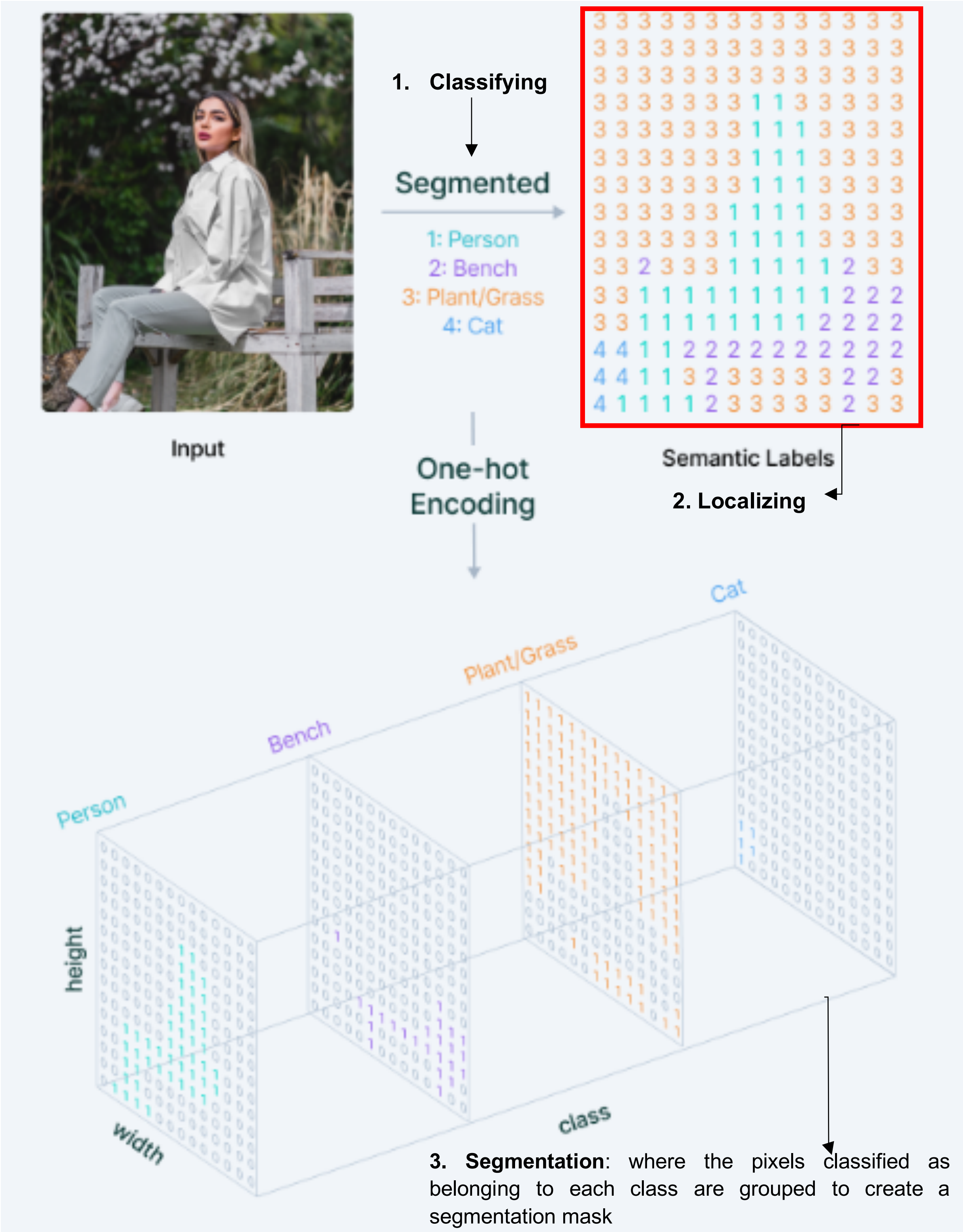


Figure 2 -Main stages and process of Semantic Image Segmentation

Source - https://www.v7labs.com/blog/semantic-segmentation-guide

1.3 Segmentation Process in this study

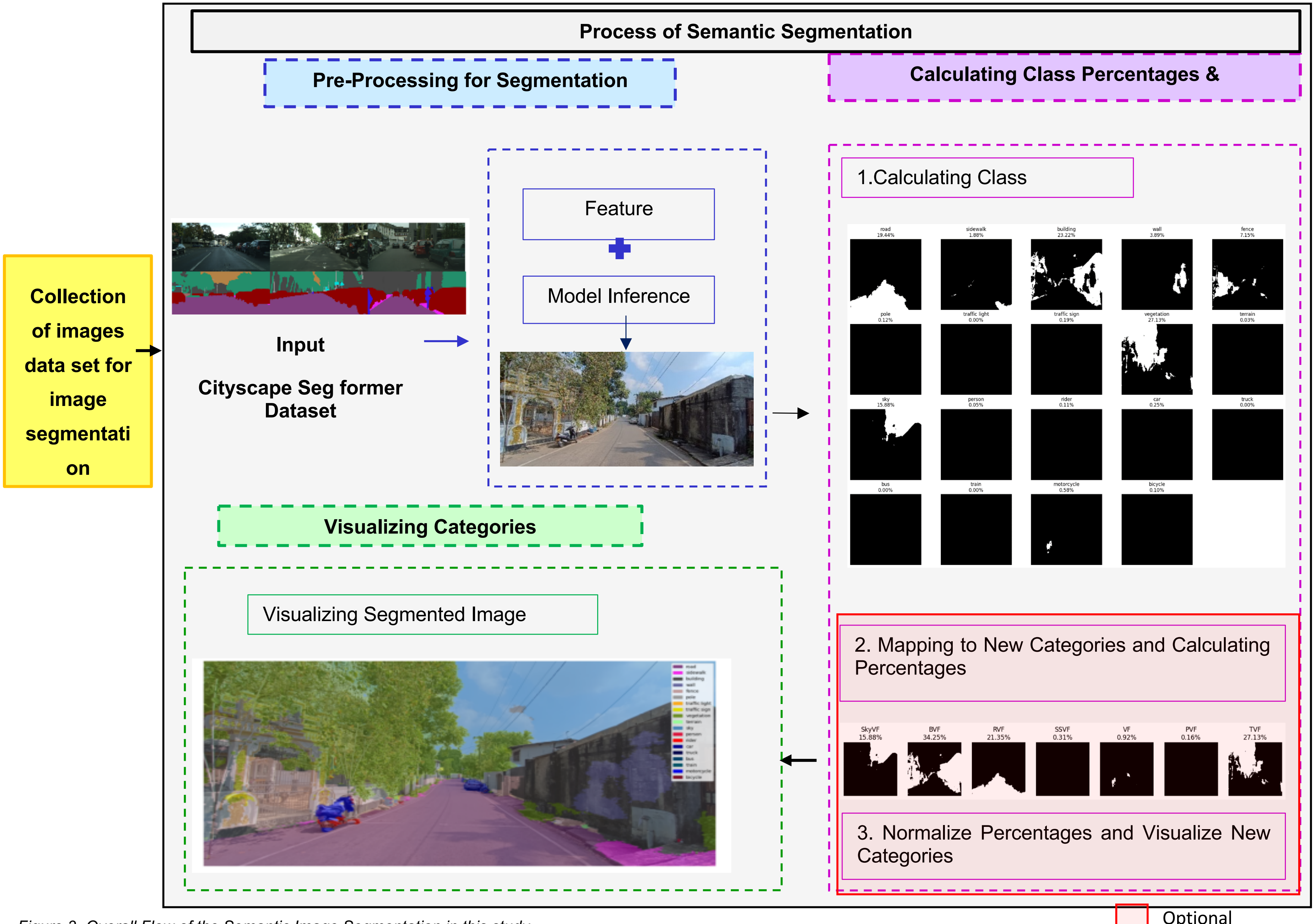


Figure 3 -Overall Flow of the Semantic Image Segmentation in this study

1.3.1 Data Collection

What does this study used?

The case study evaluated 200 manually captured environmental images in and around the University of Moratuwa using semantic segmentation.

The camera features an 8-megapixel resolution and an f/2.2 aperture, ensuring satisfactory image detail and effective light capture, respectively. Featuring a 120-degree ultrawide field of vision, this device is capable of effectively capturing wide and expansive scenes as well as group photographs. Furthermore, the camera's 1/4.0-inch sensor size and 1.12µm. The Redmi Note 12 pro mobile phone has been used in this study.

Figure 4 -Collection of Collected Images for Segmentation

1.3.2 Steps of Image Semantic Segmentation

In this study, Google Colab was utilized as the primary computing environment for running the image segmentation code.

Google Colab,

Google Colab is a cloud-based platform provided by Google that allows for the creation and sharing of Jupyter notebook files. It offers free access to GPU (Graphical Processing Unit) and TPU (Tensor Processing Unit) resources, making it ideal for running machine learning algorithms and data analysis tasks.

Visual Studio Code (VS Code) with Python Extension, PyCharm, RStudio also can use.

Figure 5 -Icon of the Google Colab

You can access to the google colab through the website.

For that you need to have a google or Gmail account.

https://colab.research.google.com/

Step 01

To begin coding, click on 'File' in the menu bar, then select 'New notebook' to create a new notebook. Once the notebook is created, you can start writing and executing code by following the provided code snippets below.

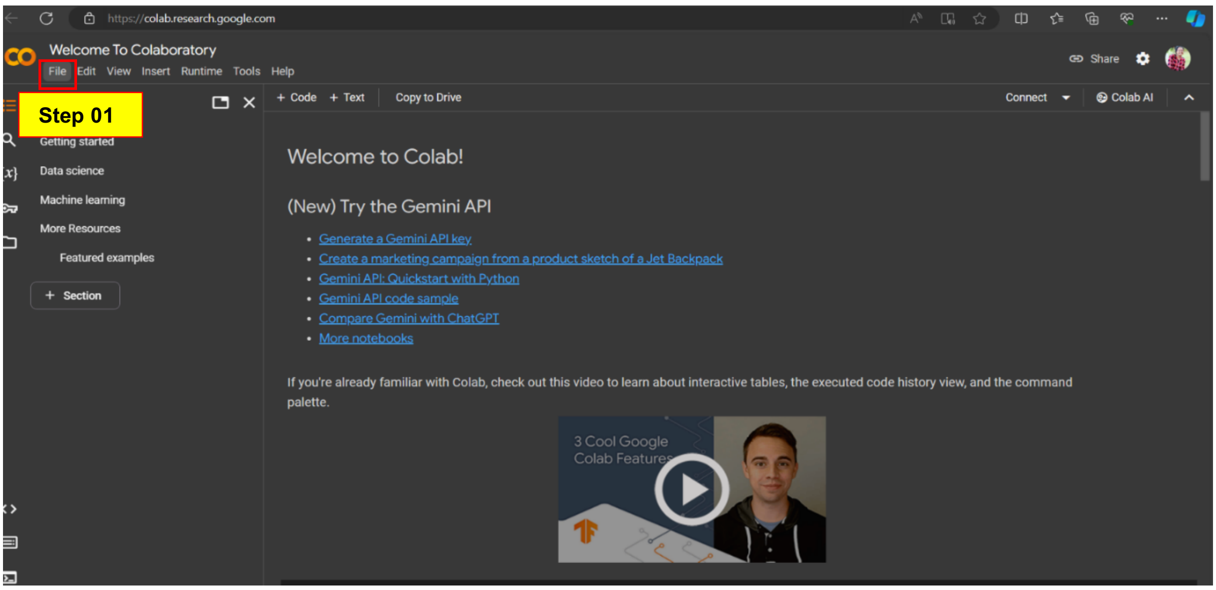


Figure 6 - Icon of the Google Colab

Editable Code (As per the purpose of the study, user need to change the code

Fixed\Common Code (User can use this code as it is)

Step 02
Before starting, students need to upload their dataset (images) to their Google Drive. Through the google drive, you can upload your images for the google colab

Step 03
Mount Google Drive

```
from google. colab import drive.

# Mount Google Drive

drive. mount('/content/drive')
```

This line imports the necessary library drive from the Google Colab package, which enables interaction with Google Drive.

Mounting Google Drive: This command mounts your Google Drive in the Colab environment. It prompts you to click on a link and authenticate your Google account to grant access to your Google Drive files.

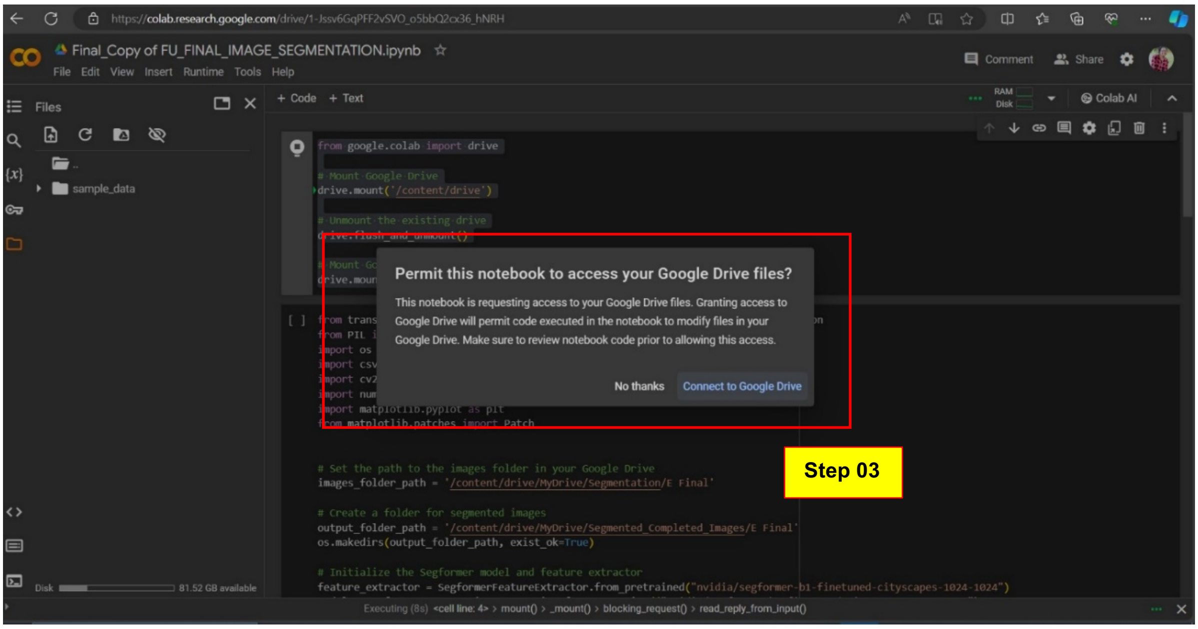


Figure 7 -Mount the Google Drive

When you run the code, as shown in figure 07 it will prompt you to connect Google Drive to Colab. Simply click on 'Connect to Google Drive' to grant permission and establish the connection.

After clicking 'Connect to Google Drive', the connection will be shown, as shown in Figure 08. If the connection does not appear, by right-clicking you can refresh the area.

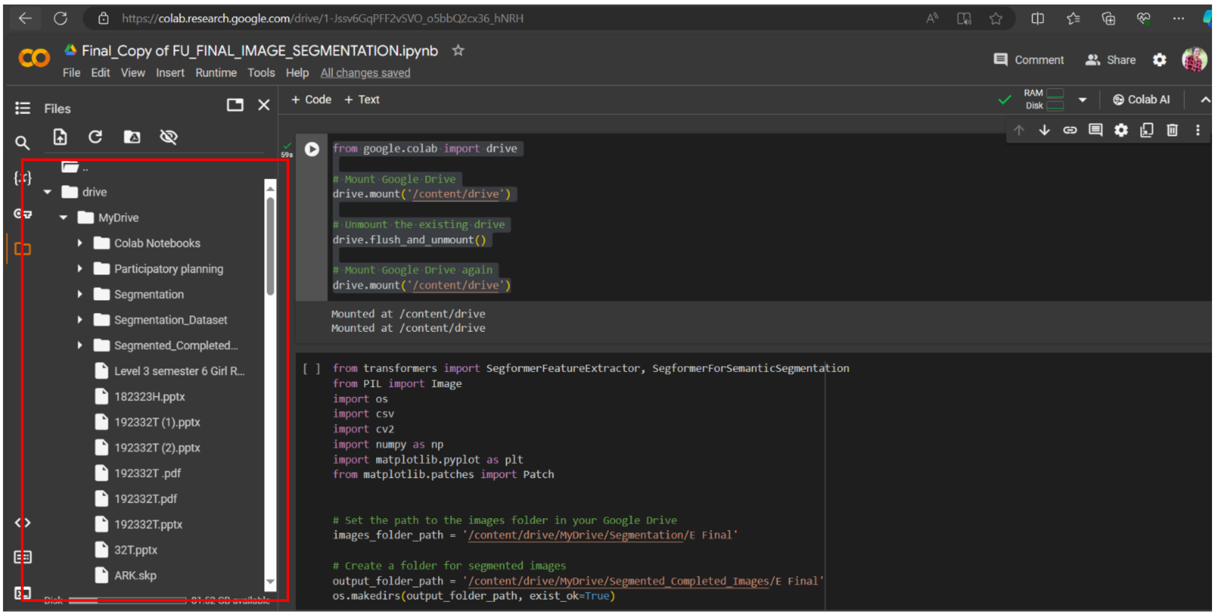


Figure 8 -Connection with Google Drive

Apart from uploading the dataset to Google Drive, students have other options for accessing data in Google Colab:

- Direct Upload: Instead of uploading the dataset to Google Drive, students can upload files directly to the Colab environment using the file upload feature. This allows them to upload files from their local system without relying on Google Drive.

- Mounting Other Cloud Storage: In addition to Google Drive, can mount other cloud storage services such as Google Cloud Storage (GCS), Dropbox, or Amazon Simple Storage Service (S3) to Colab. This provides flexibility in accessing data stored in different cloud platforms.

- External Data Sources: Can fetch data from external sources such as URLs, APIs, or databases directly within their Colab notebooks. This allows them to access real-time or dynamic data for their analysis and projects.

Step 04

Before proceeding to the next code, as illustrated in Figure 09, click on the 'Code' tab to access a new environment for typing the code.

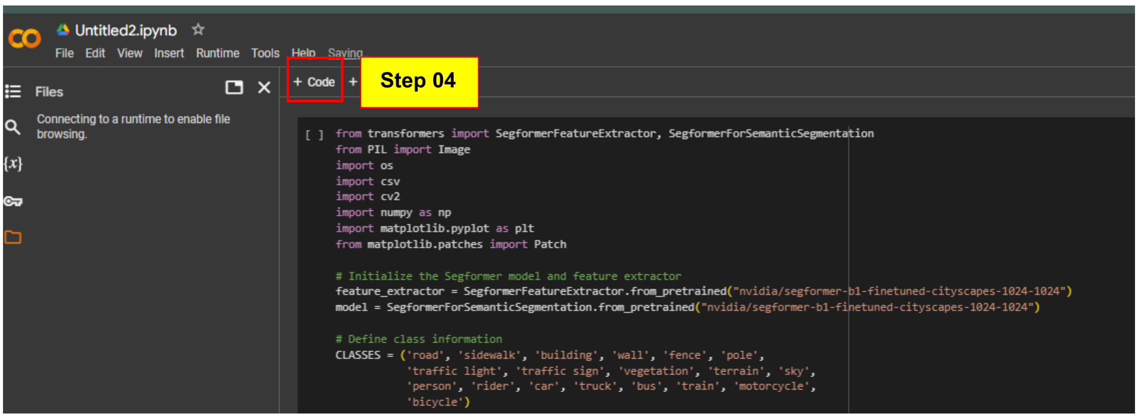


Figure 9 -Getting new code environment.

Unmount and Remount Google Drive (Optional):

If you write the previous code and the next codes together, the drive will mount repeatedly, causing errors. In such cases, you need to unmount the drive and remount it, as demonstrated in the code.

```
# Unmount the existing drive (Optional)
drive.flush_and_unmount()
# Mount Google Drive again (Optional)
drive.mount('/content/drive')
```

Step 05
Importing Required Libraries

```
from transformers import SegformerFeatureExtractor, SegformerForSemanticSegmentation
from PIL import Image
import os
import csv
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In this step, we import the necessary libraries and modules required for performing semantic segmentation and processing the data.

- **transformers:** This library provides access to pre-trained models and tools for natural language processing and computer vision tasks. We specifically import SegformerFeatureExtractor and SegformerForSemanticSegmentation for semantic segmentation.
- **PIL**: The Python Imaging Library (PIL) provides capabilities for opening, manipulating, and saving many different image file formats.
- **os:** This module provides functions for interacting with the operating system, such as accessing files and directories.
- **csv:** The CSV module provides functions for reading and writing CSV files, which may be used for storing segmentation results or other data.
- **cv2**: OpenCV is a library for computer vision and image processing tasks. We import it as cv2 to perform various image processing operations.
- **numpy:** NumPy is a library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- **matplotlib.pyplot**: Matplotlib is a plotting library for Python. We import the pyplot module to create visualizations, such as displaying segmented images.
- **matplotlib.patches:** This module provides classes for creating graphical patches, such as rectangles and circles, which may be useful for annotating images

Cityscape Segformer Library:

- Segformer is a library developed by NVIDIA for semantic segmentation tasks, which involves classifying each pixel in an image into a predefined category.
- It utilizes Transformer-based architectures, which have gained prominence in natural language processing tasks but have also shown promising results in computer vision tasks, including semantic segmentation.
- The Segformer library provides pre-trained models and tools to perform semantic segmentation on images.
- It offers various pre-trained models with different architectures and sizes, allowing users to choose a model suitable for their specific requirements, such as accuracy and computational resources.
- The library includes components for feature extraction and semantic segmentation, making it easy to integrate into machine learning pipelines for image analysis.

Why we used pre trained model?

- Pre-trained models have been trained on large datasets for extended periods, leveraging significant computational resources. By using pre-trained models, we can save time and computational resources that would otherwise be required for training from scratch.
- Pre-trained models are often trained on large, diverse datasets for general tasks such as image classification or object detection.
- Pre-trained models have already learned meaningful representations of features in images, which can generalize well to new, unseen data. Fine-tuning or adapting these models on our dataset can lead to improved performance compared to training a model from scratch.
- Many pre-trained models are trained on datasets containing a wide variety of images from different domains.
- Pre-trained models are readily available through libraries such as TensorFlow, PyTorch, or Hugging Face Transformers. This makes it easy to integrate them into our workflow and experiment with different models to find the one that best suits our needs.

There are several other pre-trained models that can be used for semantic segmentation tasks:

- DeepLabV3/V3+
- UNet:
- FCN (Fully Convolutional Network)
- PSPNet (Pyramid Scene Parsing Network
- HRNet (High-Resolution Network)
- ENet (EfficientNet)
- BiSeNet (Bilateral Segmentation Network)

Step 06

Set Paths and Initialize Models

```
images_folder_path = '/content/drive/MyDrive/Segmentation/E Final'

output_folder_path = '/content/drive/MyDrive/Segmented_Completed_Images/E Final'

os.makedirs(output_folder_path, exist_ok=True)


feature_extractor = SegformerFeatureExtractor.from_pretrained("nvidia/segformer-b1-finetuned-cityscapes-1024-1024")

model = SegformerForSemanticSegmentation.from_pretrained("nvidia/segformer-b1-finetuned-cityscapes-1024-1024")
```

We set the paths to the input images folder (images_folder_path) and the output segmented images folder (output_folder_path). The folders are created if they don't exist already.

Before running the code, it's essential to create the necessary folders and ensure that the paths specified in the code match the actual paths where your data is stored in Google Drive.

To copy the path as illustrated in the figure you must right-click in the folder and copy the path. Then paste it in the specific area in the code.

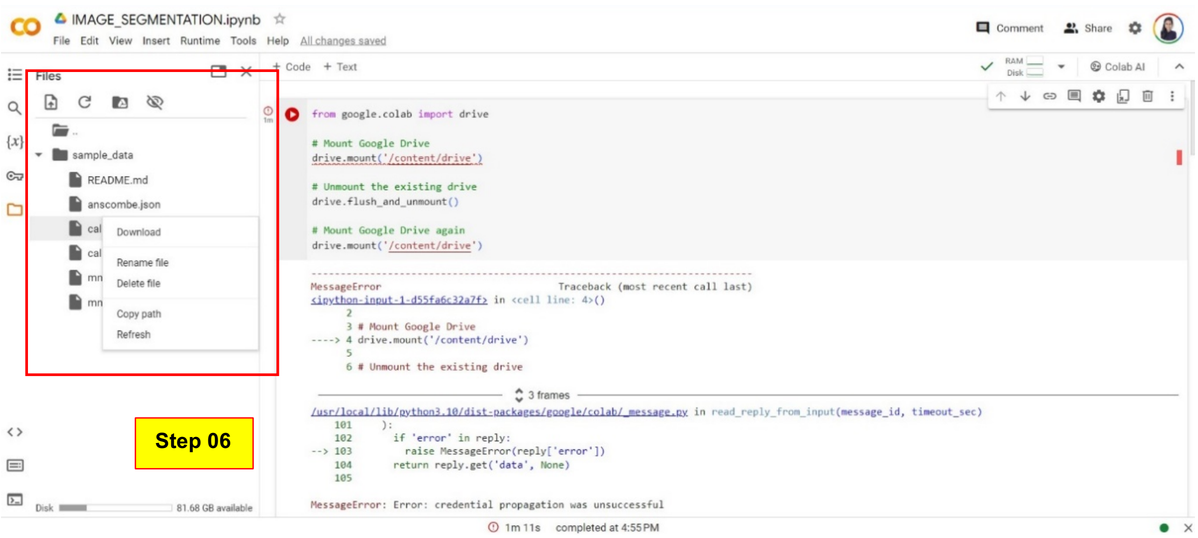


Figure 10 -Copy pasting in Google Colab

Then We initialize the Segformer model and feature extractor using pre-trained weights from the Hugging Face Transformers library.

Step 07
Define Class Information and Palette

```
CLASSES = ('road', 'sidewalk', 'building', 'wall', 'fence', 'pole',
           'traffic light', 'traffic sign', 'vegetation', 'terrain', 'sky',
           'person', 'rider', 'car', 'truck', 'bus', 'train', 'motorcycle',
           'bicycle')
palette = [[128, 64, 128], [244, 35, 232], [70, 70, 70], [102, 102, 156],
           [190, 153, 153], [153, 153, 153], [250, 170, 30], [220, 220, 0],
           [107, 142, 35], [152, 251, 152], [70, 130, 180], [220, 20, 60],
           [255, 0, 0], [0, 0, 142], [0, 0, 70], [0, 60, 100],
           [0, 80, 100], [0, 0, 230], [119, 11, 32]]
```

This section defines the classes or categories that the semantic segmentation model will predict.

The classes are typically predefined based on the specific dataset. In the provided code, the CLASSES variable contains a list of class names such as "road", "building", "person", etc., representing common categories found in urban scenes.

The palette is used for visualizing the segmented images. It assigns a unique color to each class label, making it easier to distinguish between different objects/classes in the segmented image. In the provided code, the palette variable contains a list of RGB color values corresponding to each class label. These colors will be used to colorize the segmented regions in the output images.

Visualization aids in understanding the segmentation results and assessing the model's accuracy.

Step 08
Define Category Mappings

This step is optional, and depending on the study's purpose, you can streamline the code to focus on specific aspects. The table provides the meanings of each category.

```
category_mappings = {
    "SkyVF": ["sky"],
    "BVF": ["building", "wall", "fence"],
    "RVF": ["road", "terrain", "sidewalk"],
    "SSVF": ["traffic light", "traffic sign", "pole"],
    "VF": ["car", "truck", "bus", "train", "motorcycle", "bicycle"],
    "PVF": ["person", "rider"],
    "TVF": ["vegetation"],
```

We define mappings between the original classes (CLASSES) and new categories for analysis (category_mappings). This allows us to aggregate pixel counts for specific categories in the segmentation results. Visualization aids in understanding the segmentation results and assessing the model's accuracy.

Variable	Theoretical Meaning
Sky view factor (SkyVF)	Percentage of sky pixels of the image
Building view factor (BVF)	Percentage of building pixels of the image
Road view factor (RVF)	Percentage of road pixels of the image
Sign symbol view factor (SSVF)	Percentage of sign symbol pixels of the image
Vehicle view factor (VF)	Percentage of car pixels of the image
People view factor (PVF)	Percentage of people pixels of the image
Tree view factor (TVF)	Percentage of tree pixels of the image

Table 2 - Meaning of street view factor.

Steps 09 to 14 contain default codes.

Step 09
Initialize Results List

```
results_list = []
```

We initialize an empty list (results_list) to store segmentation results for each image in the dataset.

Step 10
Loop Over Images

```
for image_name in os.listdir(images_folder_path):
    if image_name.endswith(('.jpg', '.jpeg', '.png')):
```

We loop over each image file in the input images folder. This ensures that segmentation is performed on each image individually.

Step 11
Load and Process Image

```
image_path = os.path.join(images_folder_path, image_name)
image = Image.open(image_path)
inputs = feature_extractor(images=image, return_tensors="pt")
```

We load each image using the PIL library and process it for input to the Segformer model using the feature extractor.

Step 12
Run Segmentation Model

```
outputs = model(**inputs)

logits = outputs.logitsinputs = feature_extractor(images=image, return_tensors="pt")
```

We pass the processed image through the Segformer model to obtain segmentation logits.

Step 13
Post-process Segmentation Results

```
seg = logits.argmax(dim=1)[0]

color_seg = np.zeros((seg.shape[0], seg.shape[1], 3), dtype=np.uint8)

palette = np.array(palette)

for label, color in enumerate(palette):

    color_seg[seg == label, :] = color

color_seg_resized = cv2.resize(color_seg, (image.width, image.height))

img = (np.array(image) * 0.5 + color_seg_resized * 0.5).astype(np.uint8)
```

We post-process the segmentation results by converting logits to segmentation masks and applying a color palette. The segmented image is resized to match the original image size and combined with the original image for visualization.

Step 14
Save Segmented Image

```
segmented_image_name = f"segmented_{image_name}"

segmented_image_path = os.path.join(output_folder_path, segmented_image_name)

plt.imsave(segmented_image_path, img)
```

Link to the full code –

https://colab.research.google.com/drive/18nKcjebHH-qyie5JSIE11H634wkfwwOO#scrollTo=Sxz68g2kytmv

2. SPACE SYNTAX ANALYSIS

2.1 What is Space Syntax?

Space syntax refers to a collection of methods used to analyze the arrangement of spaces and the patterns of human activity in buildings and urban environments. Additionally, it comprises a collection of theories that establish a connection between space and society. Space syntax examines the spatial distribution of individuals, their movement patterns, their ability to adjust to their surroundings, and the way they communicate about these aspects. Space syntax takes a direct way of dealing with the intricacies of spatial analysis. The analysis begins by identifying the most fundamental and widespread patterns of interaction between persons and their physical surroundings [1].

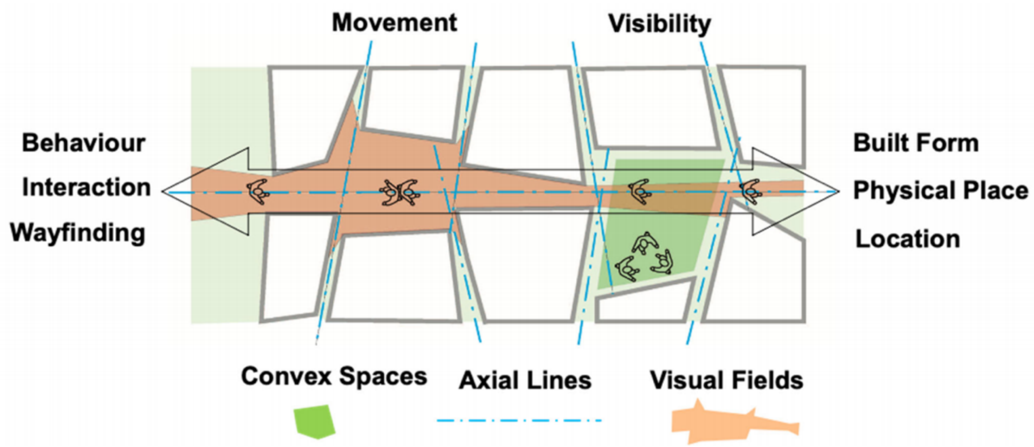


Figure 11 -Space Syntax

Source - Karimi, K. (2023). The configurational structures of social spaces: Space syntax and urban morphology in the context of analytical, evidence-based design. Land, 12(11), 2084. doi:10.3390/land12112084

The space syntax methodology employs the essential human functions of **visibility and mobility** to map the arrangement of urban environments. Individuals see their environment through their visual fields, move in linear paths, and interact with convex areas. By utilizing these principles to depict and examine spatial systems, we can create models that integrate spatial arrangement and human interactions inside a unified framework.

Space Syntax: Past, present, and future

https://www.youtube.com/watch?v=85BmaTMPQSA

1. *Karimi, K. (2023). The configurational structures of social spaces: Space syntax and urban morphology in the context of analytical, evidence-based design.* Land, *12(11), 2084. doi:10.3390/land12112084*

2.2 Steps

Step 01
Installation

Link to Install the Depth Map

https://www.spacesyntax.online/software-and-manuals/depthmap/

DepthmapX is a multi-platform software platform to performs a set of spatial network analyses designed to understand social processes within the built environment.

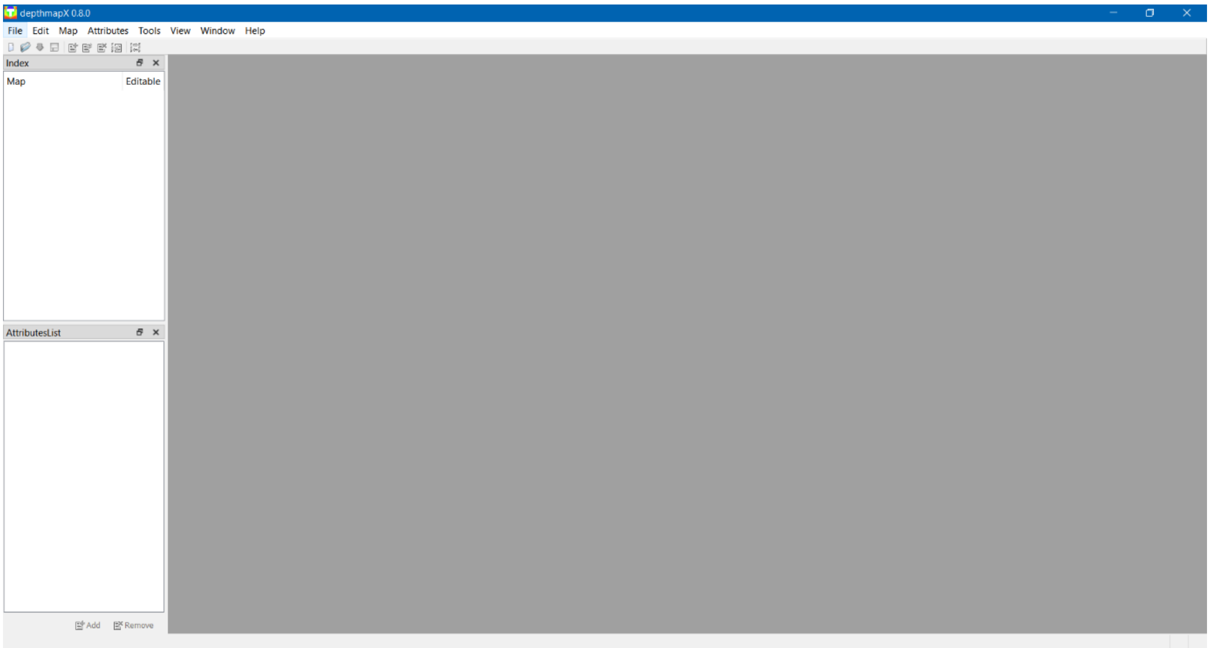


Figure 12 -Interface of the Depth map

Step 02

Click on “File > New. The new file will be created.

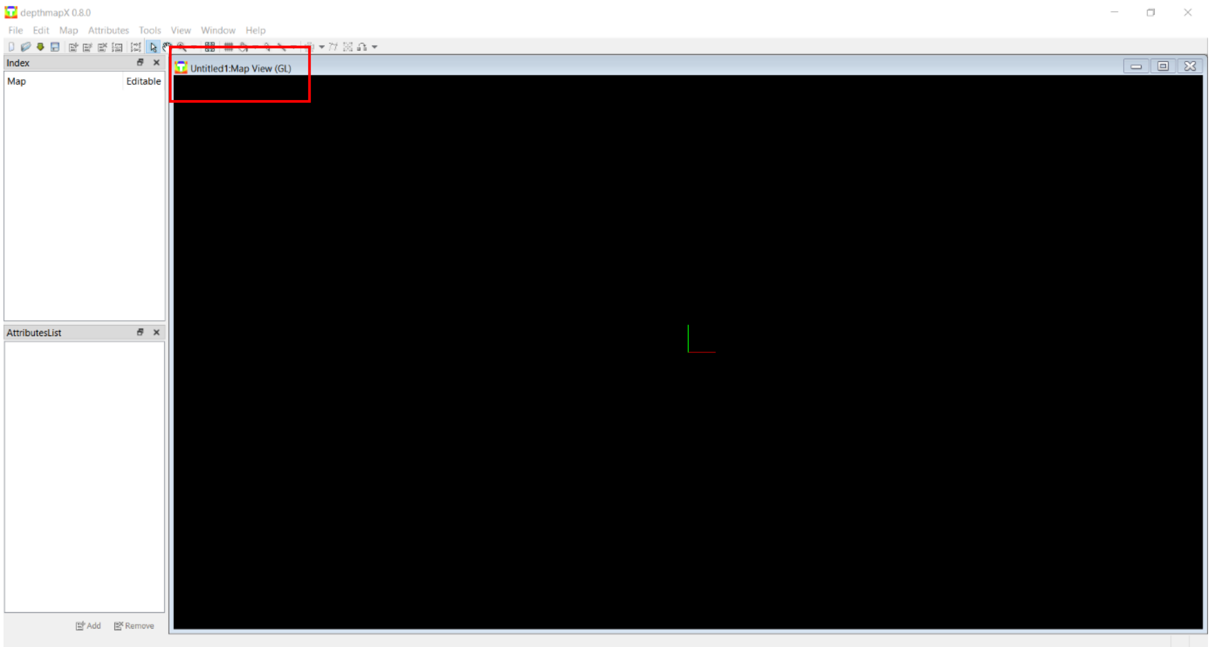


Figure 13 -Create a New Map

Step 03

Once you have a new Depth map file (. graph), you may import graphic files using the **IMPORT option under the MAP menu**. For this demonstration, please import the file.dxf. The imported graphics will be listed on the top left pane as Drawing Layers. Your graphics file will be displayed in the window on the right.

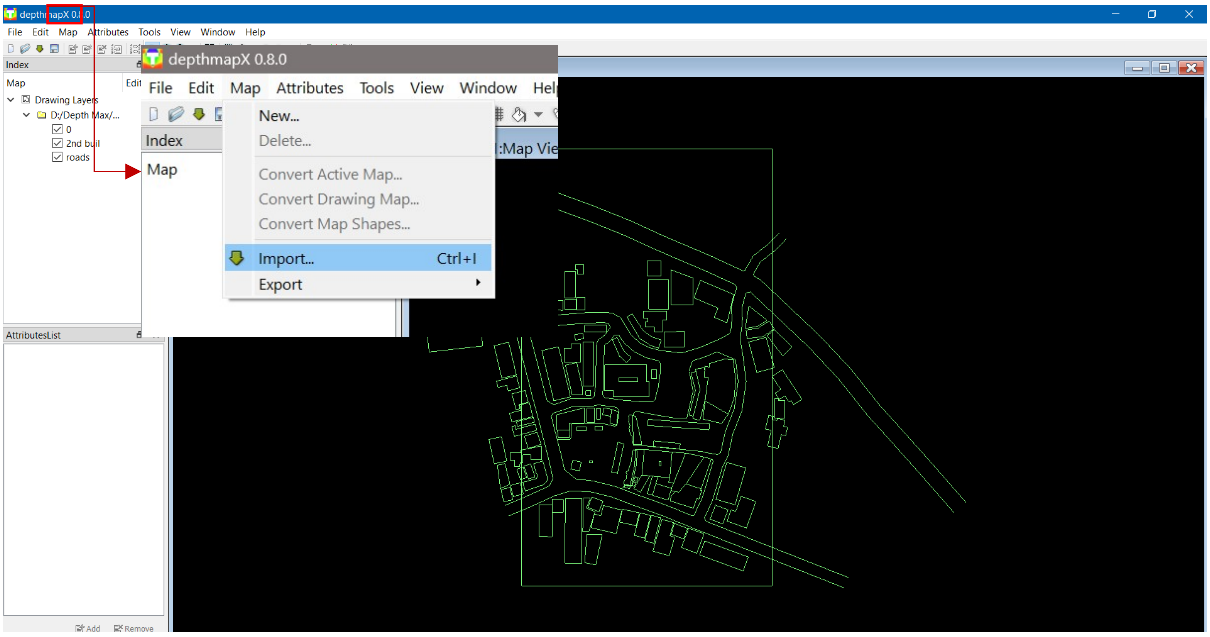


Figure 14 -Import Map

Before running the analysis, it is important to set the boundary. Here, AutoCAD was used to set the boundary, as illustrated. Alternatively, ArcGIS can also be used for this purpose. And save it as “dxf” file.

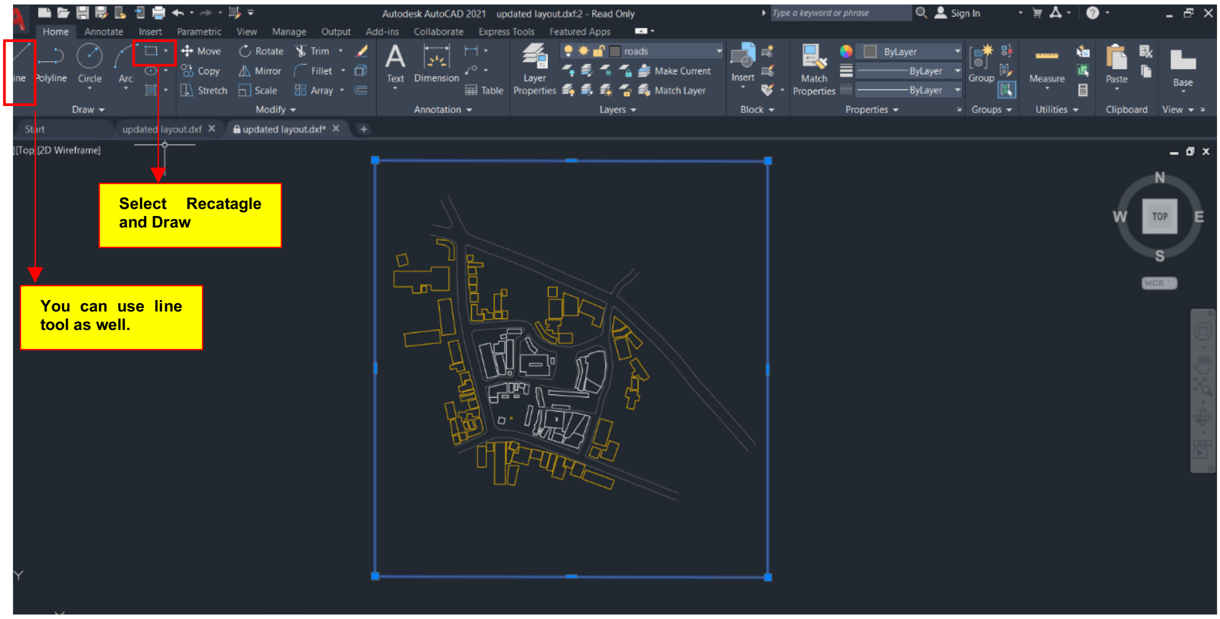


Figure 15 — AutoCAD Layer

Step 04

Convert it to an axial map: From the top menu **map, choose convert drawing map.**

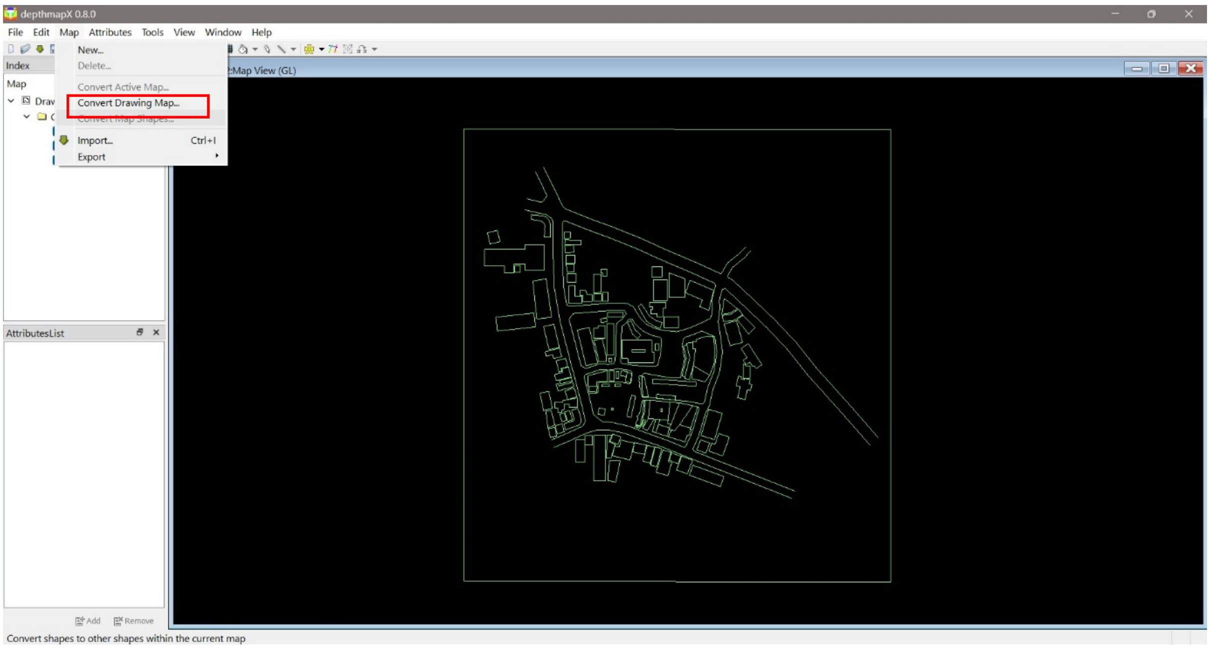


Figure 16 - Convert to Axial Map

Step 05

Choose AXIAL MAP. Click OK. Notice the new map (Axial Map) listed on the map list on the top left pane.

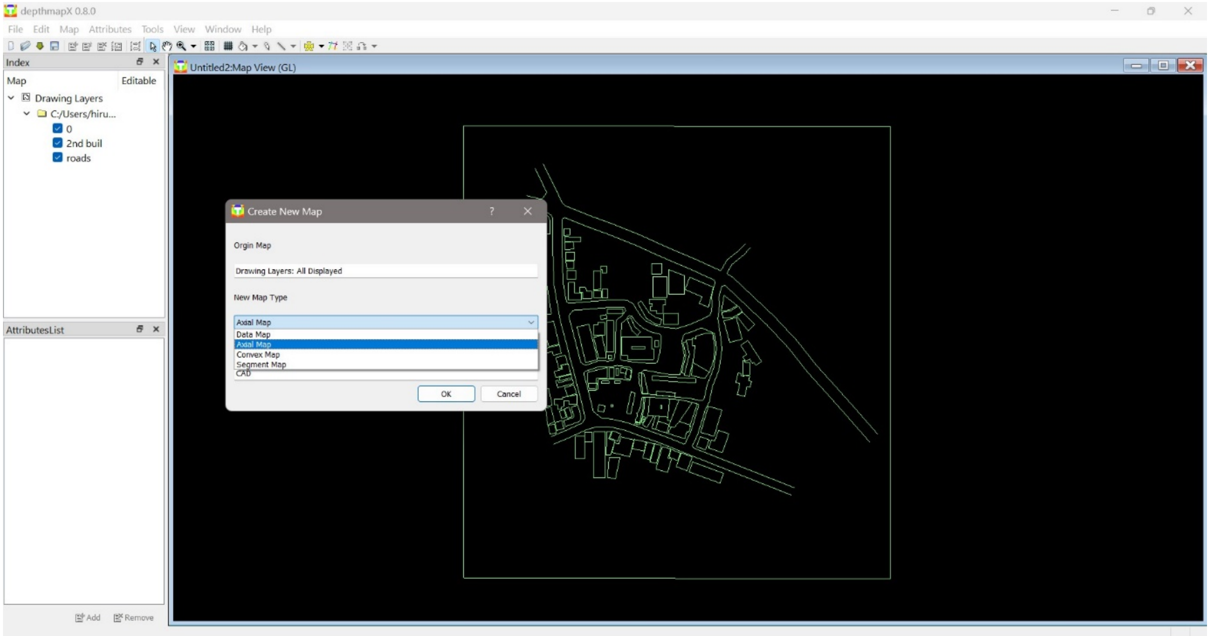


Figure 17 -Create Axial Map

Step 06

Create an all-line-axial map by clicking the  icon and clicking inside the polygon to indicate the area to be covered by the map.

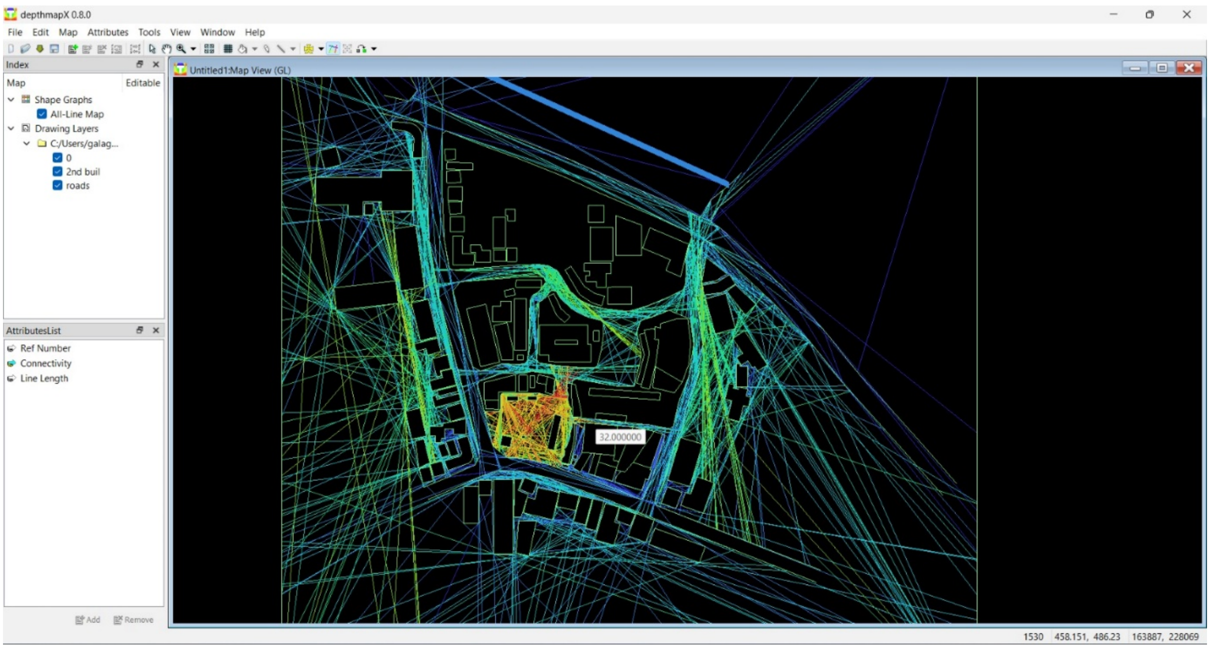


Figure 18 -Axial Map

Step 07

Then go to **the tool > Axial> Run graph analysis** as shown in the figure. Then based on your study you can get the necessary outputs. You can print directly from the Depth map. Alternatively, you can copy or export the displayed map. A depth map can export .eps and .svg file types. You can then use the image file with other applications. If you copied, the screen (EDIT, COPY SCREEN) just paste it in the desired document (text or image editors).

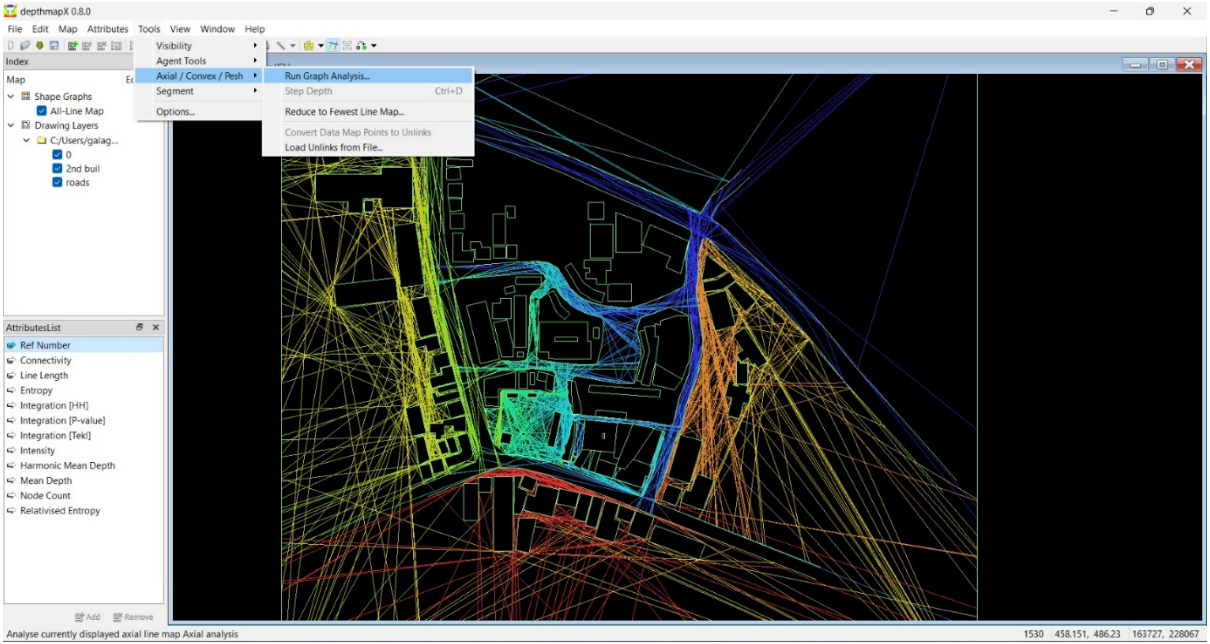


Figure 19 -Graph Analysis

Step 08

Scatterplots -Use the top menu window to choose Scatterplot.

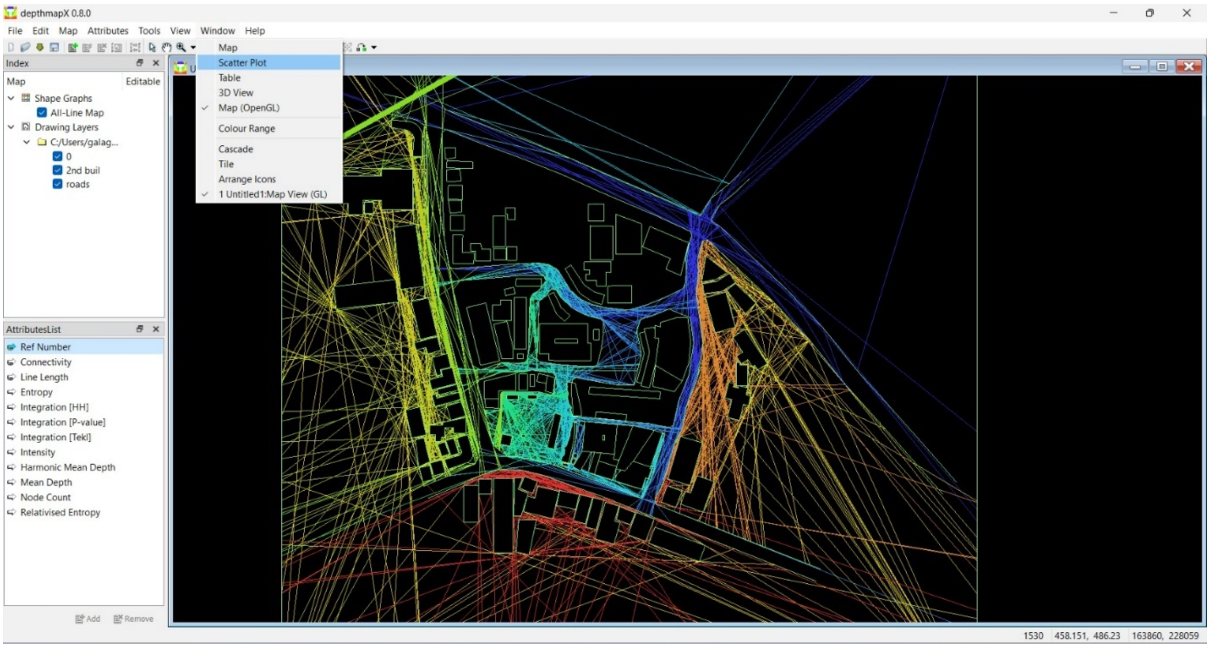


Figure 20 -Taking Scatterplots

Step 09

Next, select the required values for the X and Y fields. Here, I am preparing a scatterplot for intelligibility, where higher values indicate a greater likelihood of people visiting and staying. Therefore, for the X-axis, I have chosen connectivity, and for the Y-axis, I have selected integration.

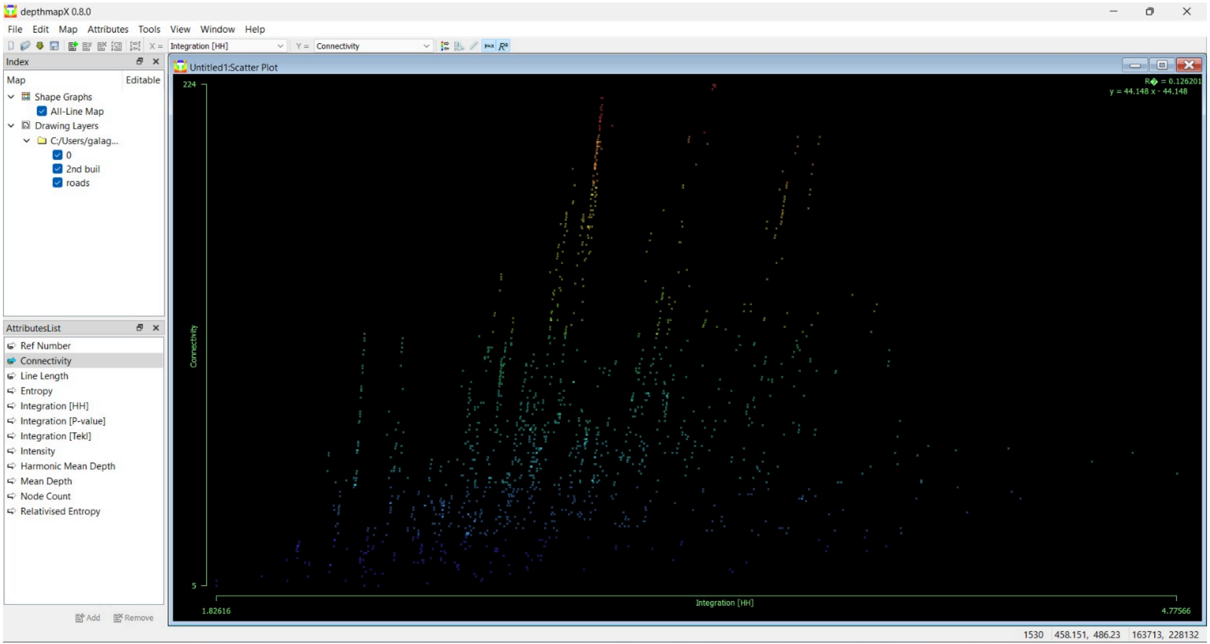


Figure 21 -Scatter Plot

Results:

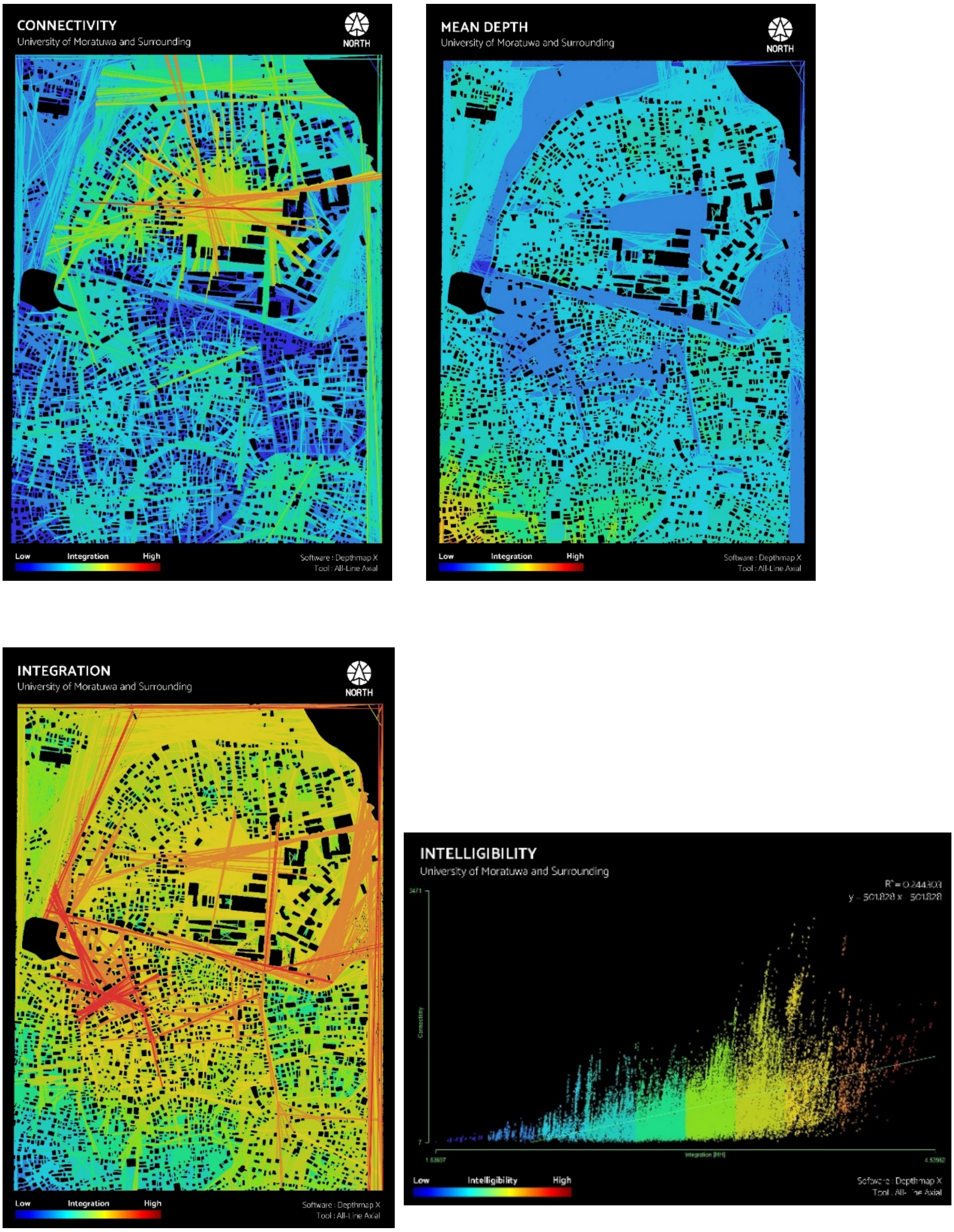


Figure 22 -Outputs of the Space syntax edited by Photoshop

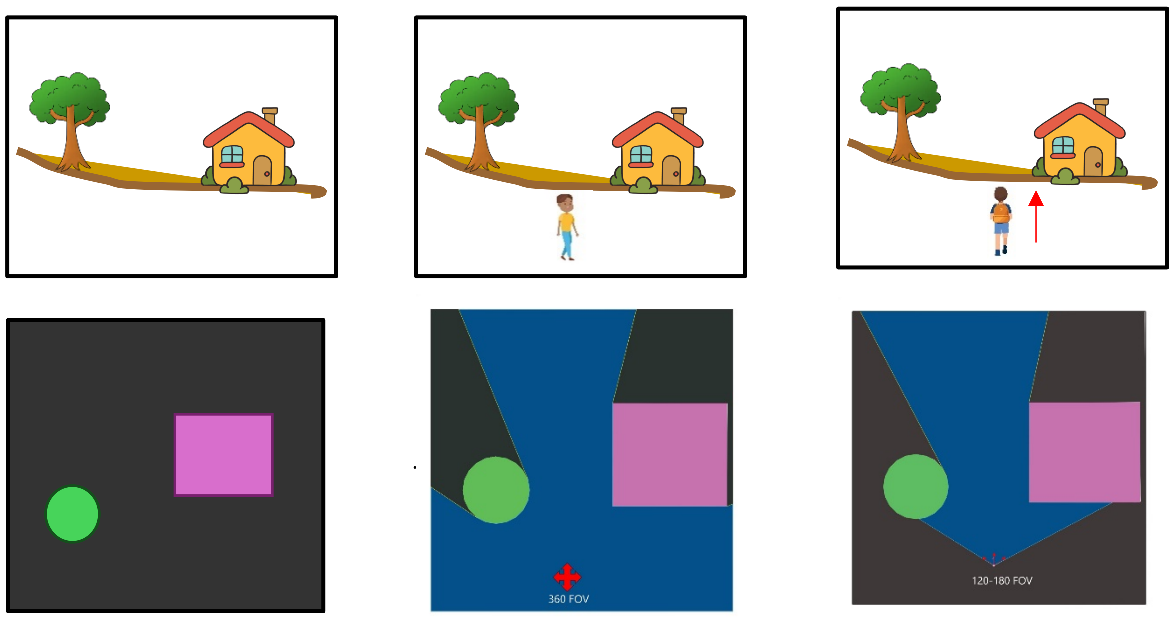
3. ISOVIST ANALYSIS

3.1 What is Isovit Analysis?

Isovists and isovist fields are conceptual tools used to carry out morphological analyses of architectural and urban spaces. Michael Benedikt (1979) defined an isovist as "**the set of all points visible from a single vantage point in space concerning an environment".** This analysis is a spatial analysis technique used in architecture, urban design, geography, and other fields to quantify the visual properties of spaces. It is an approach to describing space from the point of view of a person within an environment. It refers to the drawn polygon that covers an area that can be seen or reached when he walks in a straight line from a particular position.[2]

Isovist: An Expert Reveals

https://www.youtube.com/watch?v=Gosz3LRb-1I



(A) (B) (C)

Figure 23 -Isovist Explanation

First layer, represented by (A), identifies obstacles within the physical environment, such as walls or buildings, which may restrict visibility. The second layer, (B), delineates the potentially visible space based on both the obstacles layer and the observer's characteristics, including their location and height, encompassing a 360-degree field of view. Finally, the third layer, (C), considers contextual factors, such as the direction of view and the activity of the observer, to calculate the probable visible space in specific conditions. By integrating these layers, isovist analysis provides insights into spatial relationships and human perception, to make urban design decisions.

2.Hunter, Rebecca H.; Anderson, Lynda A.; Belza, Basia L. (2016). Community Wayfinding: Pathways to Understanding. Cham, Switzerland: Springer. p. 36. ISBN 978-3-319-31070-1.

What does this study use for Iosvit Analysis?

This study utilizes the Isovist app to analyze spatial visibility. It employs AutoCAD layers from the University of Moratuwa and its surroundings as a case study, providing valuable insights into how people perceive space in these areas.

3.2 Overview of the Isovist App

The Isovist_App is a costless software utility that may be used on multiple platforms. The Isovist_App generates a total of twenty-three fields. Out of these, ten are 'local' isovist measures that specifically pertain to the occupant's experience within the space. These measures are A**rea, Perimeter, Closed Perimeter, Compactness, Occlusivity, Vista Length, Average Radial, Drift, Variance, and Skewness** (*Benedikt, 1979).* Five Space Syntax-type metrics describe the configurational relationships over a whole design. These measures are **Choice, Mean Metric Depth, Mean Visual Depth, Mean Angular Depth, and Integration (HH).** The next eight measures are intermediate measures that incorporate both local and global information. These measures include Directed **Visibility, Co-Visibility, Overt Control, Covert Control, Counterpoint, Metric Depth to Location, Visual Depth to Location, and Angular Depth to Location.** *(Hillier & Hanson, 1984, & Turner, 2001b).*

In addition to rapid field analysis, the Isovist_App can conduct real-time justified graph diagramming, as well as point isovist, path isovist, region isovist, and isovist agent analysis. It includes a scatter plot tool that can be used to review correlations (or lack thereof) between the massive data sets produced.

Collectively, the tools of the Isovist_App allow the user to examine and isolate spatial transformations or configurative properties with bearings upon 'understanding architecture.'

Link to Isovist App

Link to Isovist Organization Platform

- https://isovists.org/2021/04/15/grid-free-integration/

Link to Isovist User Guide

- https://www.researchgate.net/publication/350107669_Isovist_App_UserGuide_v1-7

Figure 24 -Isovist App

3.3 Steps

Step 01
Initiation

> Link to Install the Iovist App
>
> https://isovists.org/

You can access the app through the provided link. Once there,

1. simply click on " login and download" and then sign up to the app and create your account.

2. Download the app according to your device

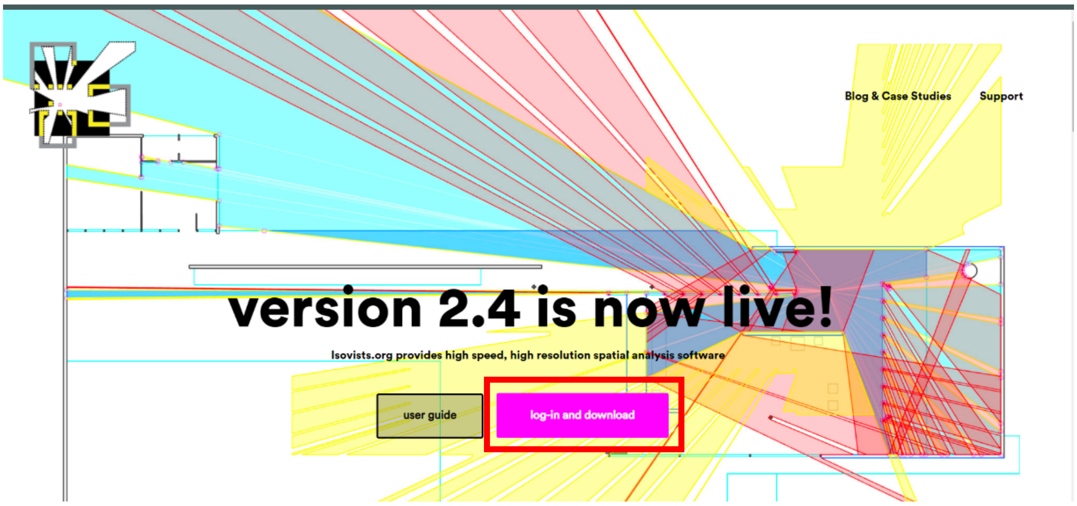


Figure 25 -Interface of the web page.

3. Then it will download the zip file and open the **isovist_2-4-exe**

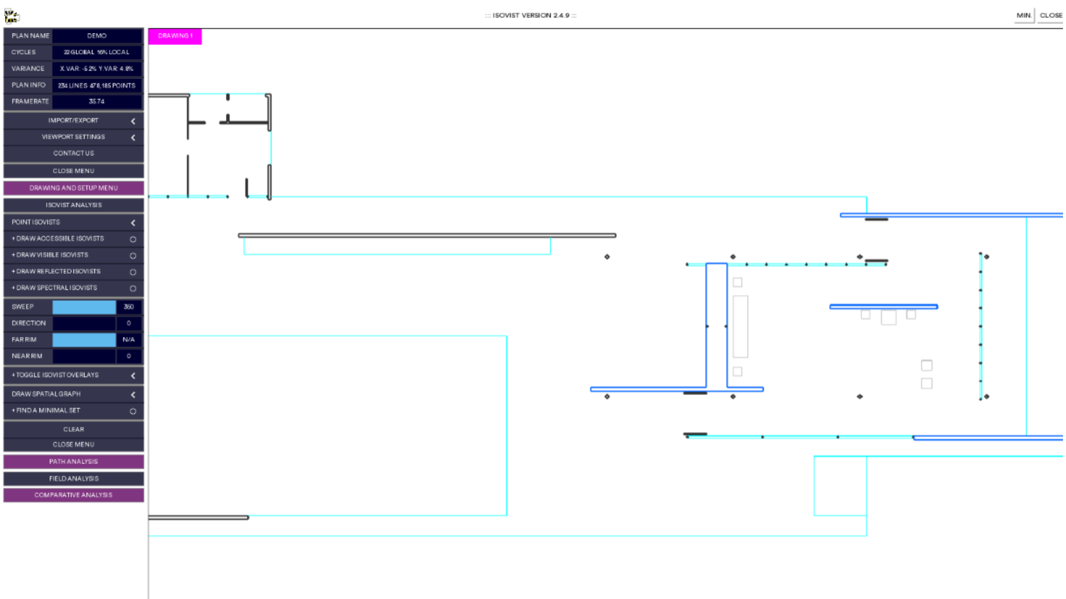


Figure 26 -Interface of the app

Step 02

Import the AutoCAD file**.**

1. Go to the "Import and Export" tab.
2. Choose "Open Scan or Plan File."
3. Browse to select the AutoCAD layer file.
4. Then the AutoCAD layer will open

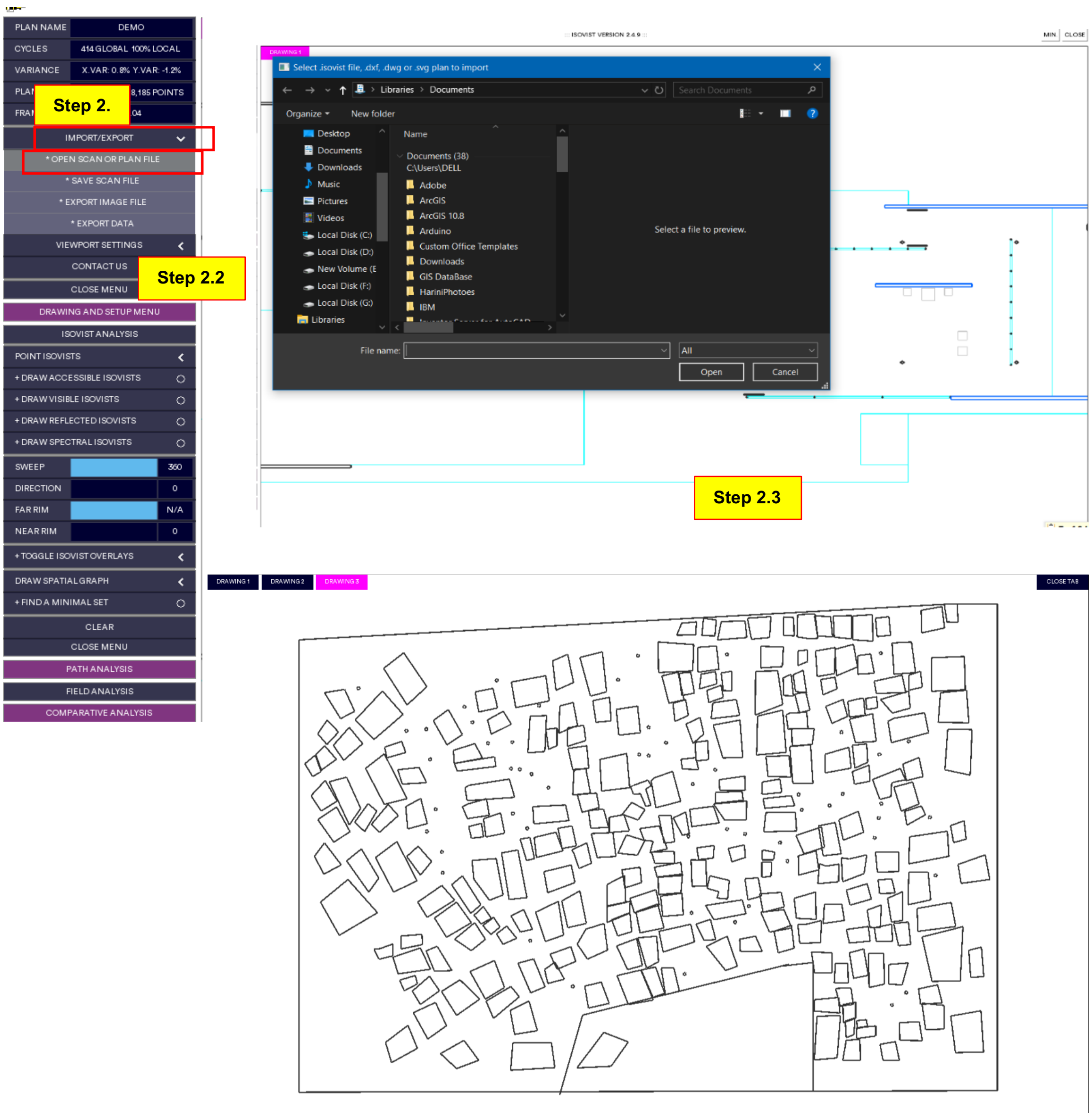


Figure 27 - Importing

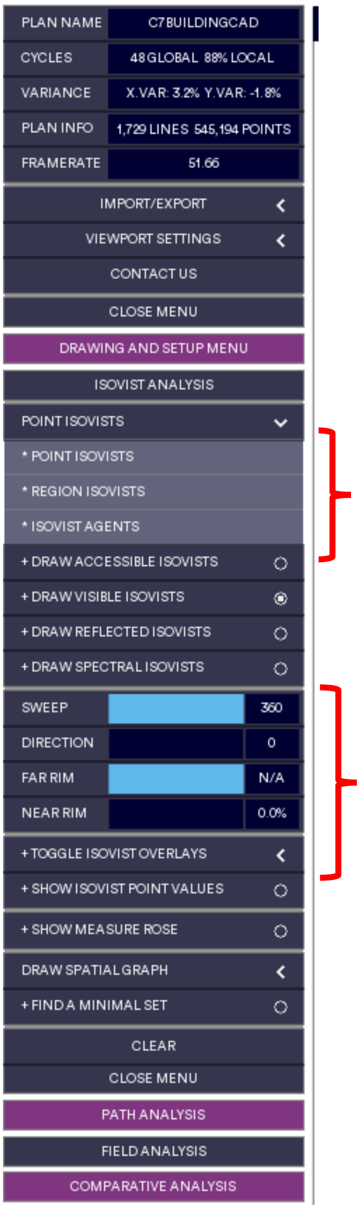
Step 2.4

> **Note -**
>
> **It's important to note that in AutoCAD, you must draw all boundaries and obstacles within the environment (Refer to the 15), as we are analyzing the visible area from specific points. Additionally, draw the buildings using lines to ensure accurate analysis.**

Step 03

Start the Analysis

Here, the analysis can vary depending on the specific purpose of the study.



- ‘Point isovists’ can be calculated and released by the user moving the cursor and clicking at a point of interest.
- ‘Region isovists’ can be calculated by the user clicking and dragging the cursor. These identify all space that is visible from along the route that the cursor is dragged along.
- ‘Isovist agents’ can be released by the user clicking the cursor. These subsequently move independently, that is, they move randomly towards available depth.

- **Isovist sweep (angle of view).**
- **Direction (heading that the isovist is ‘looking’ in).**
- **Far rim (how far the isovist can ‘see’).**
- **Near rim (an internal horizon that sets nearest visible isovist edges).**

Setting a parameter affects all latter field analysis calculations and automatically resets all analysis.

Figure 28 -Basic options in the app

You can enable or customize the analysis options according to the specific purposes of the study.

For this study, the analysis primarily focuses on studying all visible points that observers can observe during their walk. Therefore, the option

1. "Draw visible isovist" is enabled.
2. Since people typically have a field of view of around 120 to 180 degrees while walking, the sweep is set accordingly. You need to change the blue tab by dragging it.
3. The walking direction has been selected for analysis.
4. Importantly, if you are comparing different points, the option "show isovist point value" is enabled, which displays the values at the end of the screen.

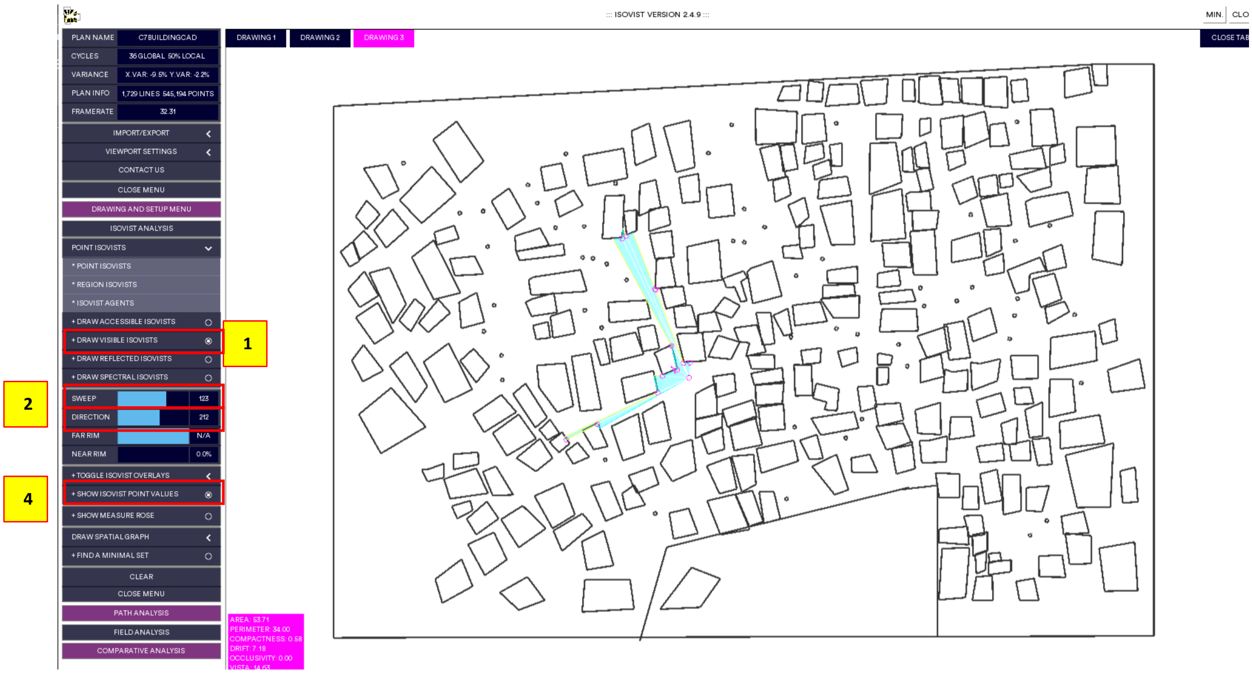


Figure 29 - Isovist app options

Results:

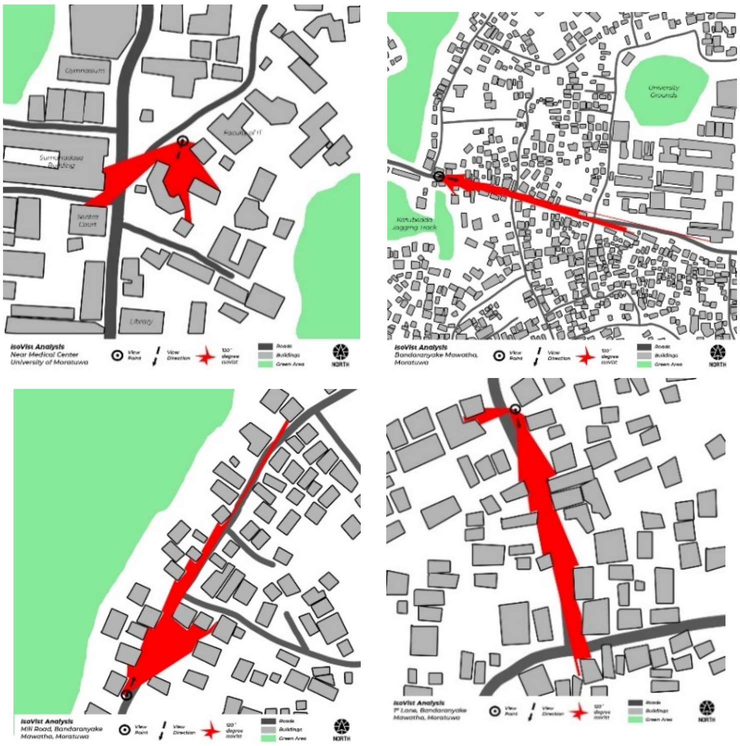
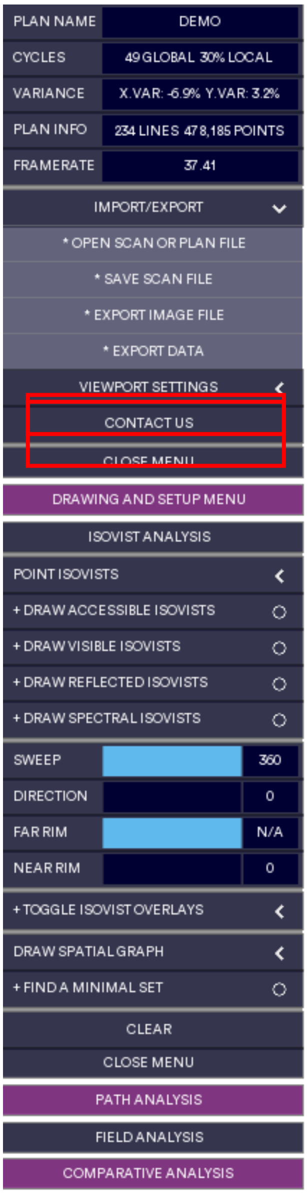
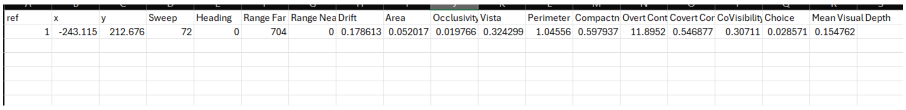


Figure 30 -Outputs of the Isovist edited by Photoshop.

Step 04
Export Data

Go to the "Import/Export" tab and choose "Export Image File" to export the image. Then, enable the "Export Data" option to save the CSV file containing the values.





ref	x	y	Sweep	Heading	Range Far	Range Nea	Drift	Area	Occlusivity	Vista	Perimeter	Compactn	Overt Cont	Covert Cor	CoVisibility	Choice	Mean Visual Depth
1	-243.115	212.676	72	0	704	0	0.178613	0.052017	0.019766	0.324299	1.04556	0.597937	11.8952	0.546877	0.30711	0.028571	0.154762

Figure 31 -Exporting

3.2 Definition of Isovist Measures

VARIABLE	THEORETICAL MEANING
Isovist area	All space visible from a specific viewpoint
Isovist perimeter	Boundary of the visible area from a specific point.
Compactness	How concentrated or spread the visible area is.
Vista	Maximum panoramic perspective at the point
Occlusivity	The proposition of edges that are not physically defined
Choice	Quantity of routes accessible within the visible area
Drift	Visual Flow within Space
Vista Length	Longest single view available at each location
Average Radial	Mean view length of all space visible from a location
Variance	Square of deviation between all radial lengths and the average radial length of an isovist
Skewness	Cube of deviation between all radial lengths and average radial length of an isovist

Table 3 - Definitions of main isovist parameters