AN ILLUSTRATED, HANDS-ON GUIDE

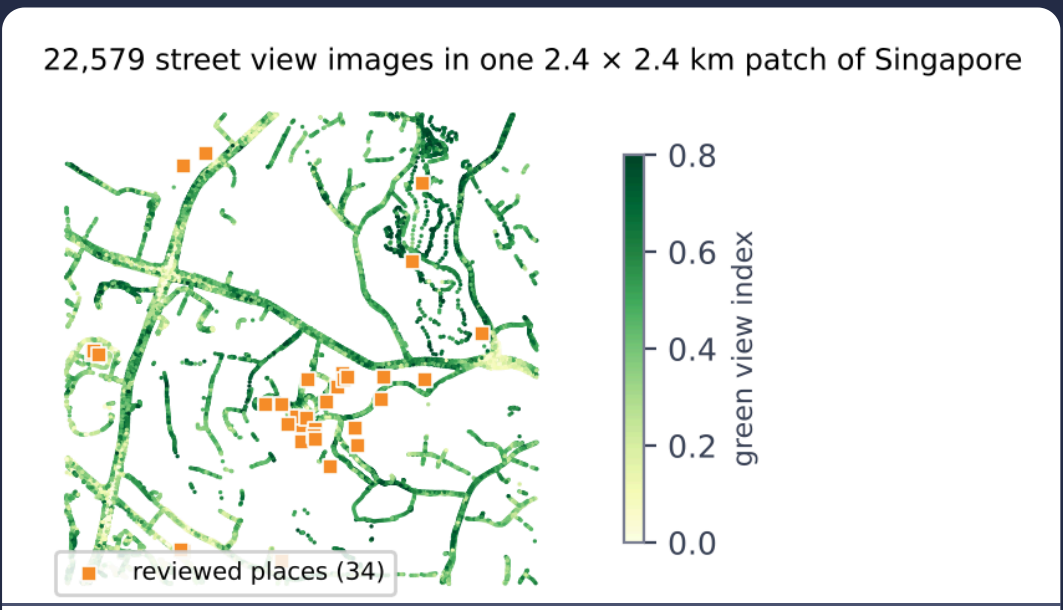
Street View Imagery for Cities

A friendly guide to reading streets from pictures, with ZenSVI and real Singapore images

22,579 labelled images from Global Streetscapes, six labs you can run anywhere, embeddings and similarity search, and a ZenSVI pipeline that downloads, segments and scores your own images.

Eight short chapters · Six labs + one pipeline · Three challenges · Cheat sheet

Prompted and directed by Mahmoud A.
Written and illustrated by Claude (Anthropic)



Prepared for the 2026 CCS: Urban AI + Agent Workshop · Harbin Institute of Technology · November 2026

How to Use This Guide

Two parts: labels you can analyse now, images you fetch yourself.

A street view image is a photo taken from a street, usually by a car, a bike or a person with a phone. Millions of them exist on open platforms. With computer vision, every image becomes numbers: how much of it is tree, sky, building, road; what kind of scene it is; how beautiful or safe it looks. This guide teaches that pipeline with **ZenSVI**, an open-source Python package from the NUS Urban Analytics Lab.

YOU WILL BE ABLE TO

- Explain how images become street indicators.
- Compute and map the green view index.
- Relate what is in a street to how it is perceived.
- Check model-made labels before trusting them.
- Find similar streets with embeddings.
- Join street images with place reviews.
- Run ZenSVI on your own area.

WHAT YOU NEED

- **Part A (Labs 1-6):** Python with `pandas`, `scikit-learn`; files `svi_lab.py`, `sg_svi.csv`; for Lab 5 also the reviews sample.
- **Part B (pipeline):** `pip install zensvi`, a free Mapillary token, `zensvi_lab.py`, 10–30 minutes.

CONTENTS

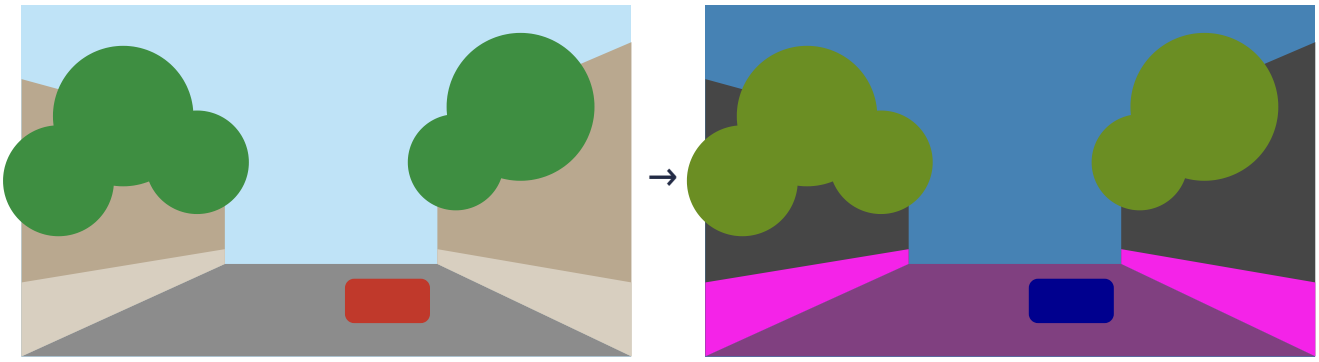
1. From pictures to numbers
2. The data · Lab 1
3. Green and sky · Lab 2
4. Perception · Lab 3
5. Check the labels · Lab 4
6. Embeddings and similarity · Lab 5
7. Reviews meet streets · Lab 6
8. ZenSVI on your laptop · Part B

About the data. The Singapore subset of *Global Streetscapes* (Hou et al., 2024; CC BY-SA 4.0): 22,579 images from Mapillary and KartaView, with labels computed by the dataset authors. In this release, all Singapore images fall in one 2.4 × 2.4 km patch around the Botanic Gardens, Tanglin and Holland Road. Every number in Part A comes from running `svi_lab.py`. Part B was written against ZenSVI 0.12 but needs your own token and images, so its results are yours to discover.

1 From Pictures to Numbers

Download, segment, classify, perceive, map.

A street view image looks at a street the way a pedestrian does. That is its strength: satellites see roofs and treetops; street images see what people see. To use thousands of images, a model reads every pixel and gives it a class. The share of pixels per class becomes a set of indicators for that point on the street.



a street (drawn for this guide) → its segmentation: one class per pixel

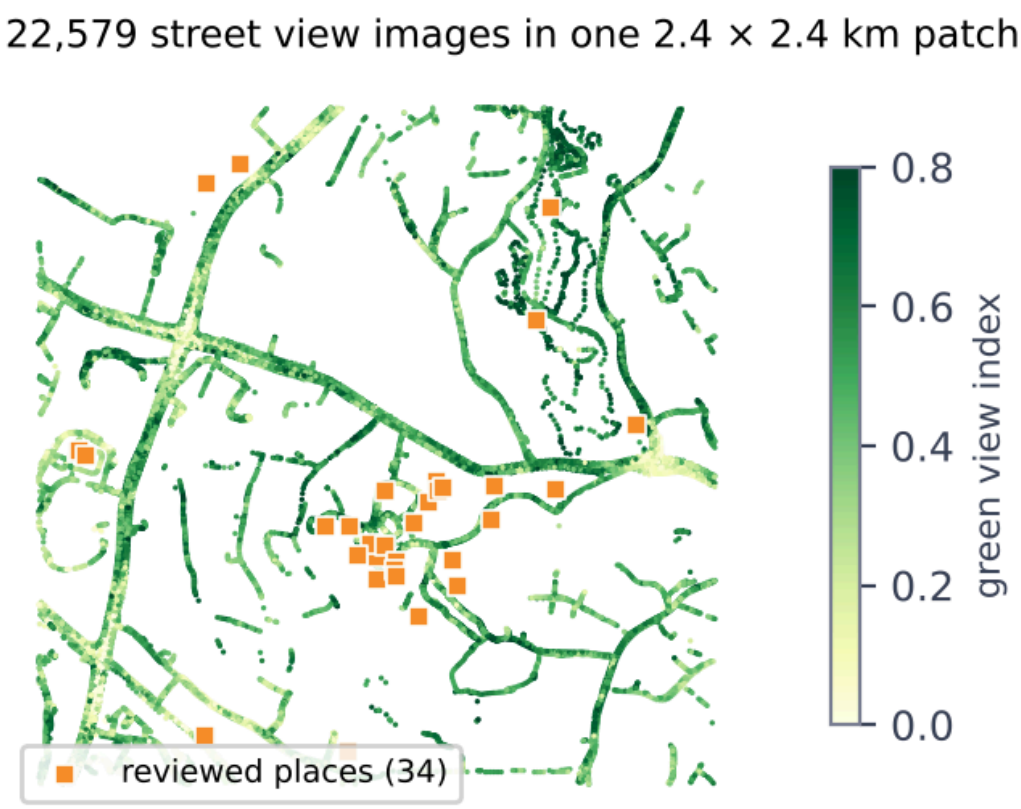


Indicator	How it is made	Used for
Green view index	share of vegetation + terrain pixels	street greenery, shade, wellbeing
Sky view index	share of sky pixels	openness, heat exposure, canyon effect
Perception scores	model trained on people’s pairwise votes	how beautiful, safe or lively a street looks
Scene and context	classifiers: scene, weather, quality, glare	filtering and describing images

2 The Data

Lab 1 · one Singapore patch, two platforms, many years.

Global Streetscapes collected 10 million open images from 688 cities and ran the same models on all of them. We use its Singapore rows, saved as one table, `sg_svi.csv`.




```
svi = pd.read_csv('sg_svi.csv')     # one row per image: lat, lon, year, source,
                                    # 16 pixel shares, 6 perception scores, scene, quality ...
svi.green_view_index.describe()     # share of the image that is vegetation + terrain
svi.groupby('scene').green_view_index.mean()
```

```
images: 22579 | sources: {'Mapillary': 12602, 'KartaView': 9977}
years: {2005: 1, 2008: 1, 2015: 2953, 2017: 8059, 2018: 495, 2019: 9049, 2020: 968, 2021: 836,
2022: 96, 2023: 121}
platform: {'driving surface': 13522, 'walking surface': 5246, 'cycling surface': 3746, 'railway':
39, 'fields': 26}
weather (predicted by a model): {'clear': 14375, 'cloudy': 7491, 'snowy': 480, 'rainy': 232,
'foggy': 1}
top scenes: {'residential_neighborhood': 4579, 'forest_road': 4032, 'highway': 1842,
'parking_lot': 1645, 'street': 973, 'parking_garage': 730}
```

WATCH OUT

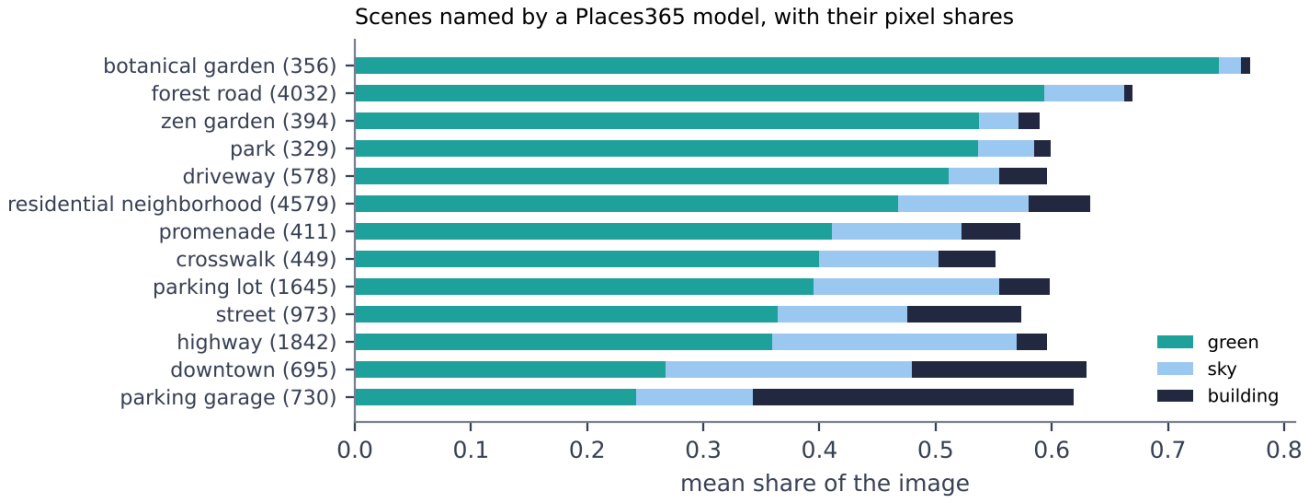
The images were taken over many years (most in 2015–2019), by different people, from cars, bikes and on foot. A street photographed in 2015 may look different today. Always record **when** and **from where** an image was taken.

3 Green and Sky

Lab 2 · the two most used street indicators.

The **green view index** (Li et al., 2015) is the share of an image that is vegetation. It measures the greenery a pedestrian actually sees, which can differ a lot from the tree canopy seen from above. The **sky view index** is the share of sky; low sky view means a closed street canyon.

```
green view index: mean 0.44 | median 0.45 | sky view: mean 0.11
                           n  green   sky  building
scene
botanical_garden           356   0.74  0.02      0.01
forest_road               4032   0.59  0.07      0.01
zen_garden                 394   0.54  0.03      0.02
park                       329   0.54  0.05      0.01
driveway                   578   0.51  0.04      0.04
residential_neighborhood  4579   0.47  0.11      0.05
promenade                  411   0.41  0.11      0.05
crosswalk                  449   0.40  0.10      0.05
parking_lot               1645   0.39  0.16      0.04
street                     973   0.36  0.11      0.10
highway                   1842   0.36  0.21      0.03
downtown                   695   0.27  0.21      0.15
parking_garage             730   0.24  0.10      0.28
the Singapore images cover one patch of about 2.4 x 2.4 km: lat 1.296-1.318, lon 103.799-103.821
```



This patch is very green (mean 0.44) because it contains the Botanic Gardens and leafy residential roads. Downtown scenes show less than half the green (0.27 against 0.59) and three times the sky of forest roads. A parking garage is mostly building.

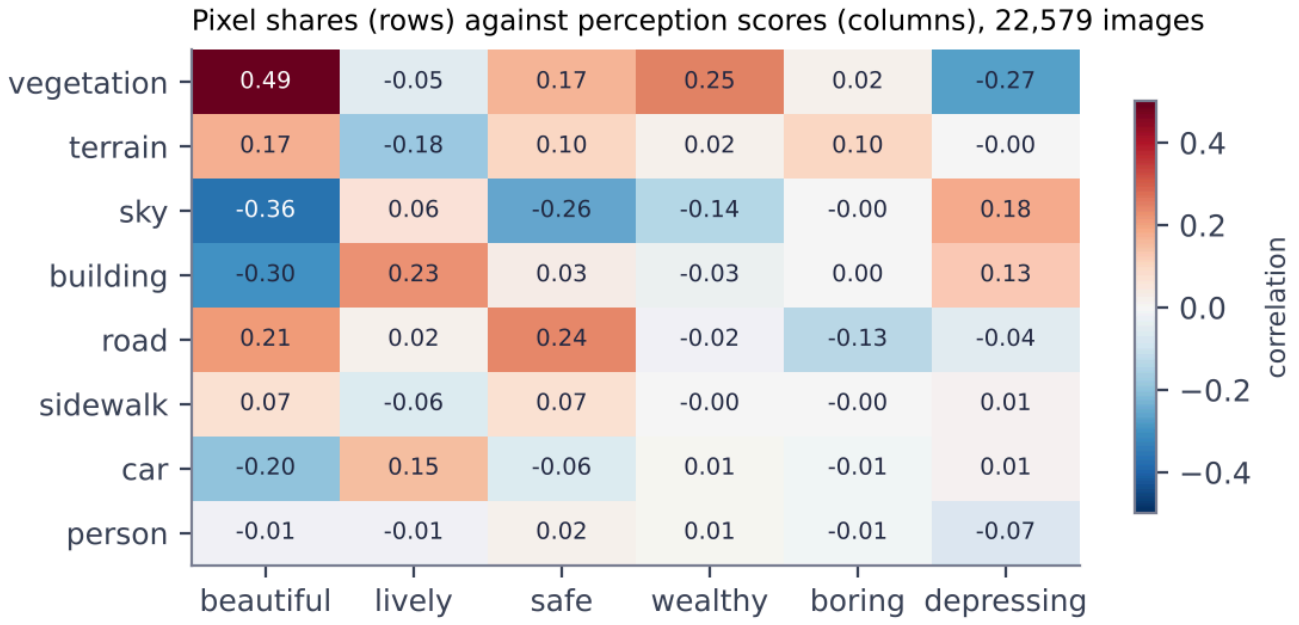
TRY THIS · EXERCISE 1

1. Map `sky_view_index` instead of green. Where are the open streets?
2. Compare images taken in 2015 with those from 2019 on the same streets. Did the green view change, or did the photographers change?

4 Perception

Lab 3 · what makes a street look beautiful (to a model)?

Perception scores come from models trained on **Place Pulse**, where thousands of volunteers chose which of two street images looked safer, livelier or more beautiful (Dubey et al., 2016). The model learned to copy those votes. Here we ask the reverse question: which pixel shares go with high scores?




```
X, y = svi[SEG], svi['beautiful']
groups = grid_groups(svi)           # 2 km blocks: test streets are not next to training streets
cross_val_score(LinearRegression(), X, y, cv=GroupKFold(5), groups=groups, scoring='r2')
cross_val_score(RandomForestRegressor(n_estimators=120, min_samples_leaf=5), X, y,
                cv=GroupKFold(5), groups=groups, scoring='r2')
```

```
from 16 pixel shares | linear 0.32 | random forest 0.33
safe     : R2 from 16 pixel shares | linear 0.10 | random forest 0.14
lively   : R2 from 16 pixel shares | linear 0.05 | random forest 0.25
```

Green goes with “beautiful” (0.49) and sky against it (−0.36). Pixel shares explain about a third of “beautiful” but little of “safe” and “lively”: those scores depend on things the shares do not capture, such as shop fronts, people and the state of buildings. Note the **spatial blocks** in cross-validation: without them, neighbouring images of the same street leak between training and test.

WATCH OUT

Do not read the linear weights one by one. The pixel shares add up to about 1, so the weights trade off against each other and can take strange values (road +9.9, bicycle −8.2). Correlations and feature importance are safer to explain.

5 Check the Labels

Lab 4 · models make mistakes, and filters change answers.

Every label in this dataset was made by a model, and models make mistakes. Before you analyse, count the impossible and the suspicious.

```
"snowy" images in Singapore: 480 | "slightly poor" quality: 30% | glare: 1.3%
  all images                       n= 22579 | green 0.439 | sky 0.114 | beautiful 6.50
  good quality only                n= 15881 | green 0.393 | sky 0.138 | beautiful 6.18
  good quality, front/back view    n= 10227 | green 0.415 | sky 0.142 | beautiful 6.57
  driving surface only             n= 13522 | green 0.427 | sky 0.147 | beautiful 6.18
  walking surface only             n=  5246 | green 0.482 | sky 0.050 | beautiful 6.69
```

The weather model says 480 images in Singapore are **snowy**. It has never snowed there: bright overcast skies or white surfaces fooled the model. 30% of images are rated slightly poor quality. Filtering changes the answers: the mean green view moves from 0.44 (all) to 0.39 (good quality only), and images taken on walking surfaces see much less sky (0.05) than those from roads (0.15), because footpaths sit under trees.

IN ONE LINE

A street indicator depends on which images you keep. Choose your filters before you look at the results, and report them.

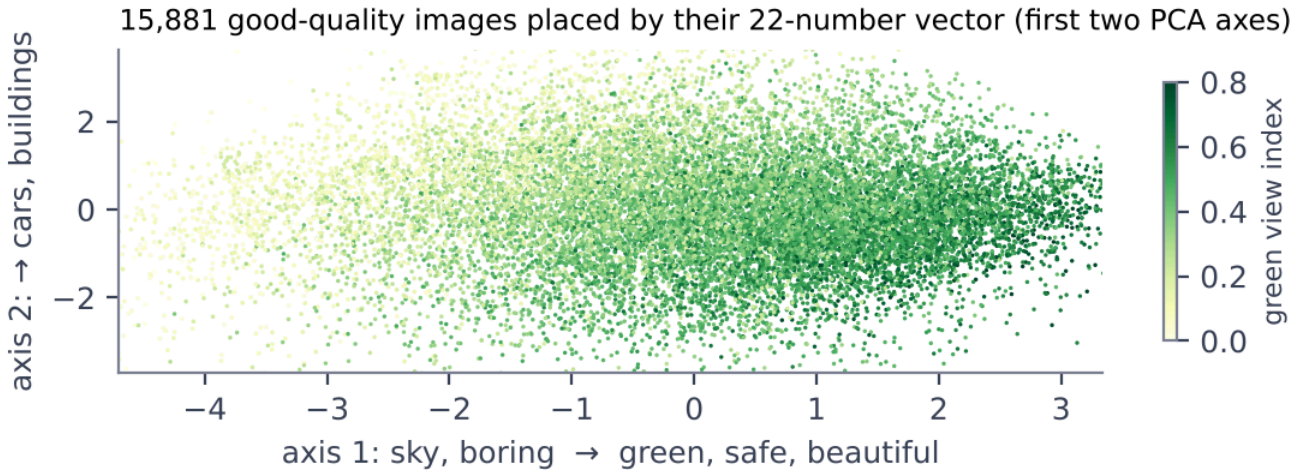
TRY THIS · EXERCISE 2

1. Look at the `scene` of the “snowy” images. What do they have in common?
2. Keep only `view_direction == "side"`. Why might side views show more building and less road?

6 Embeddings and Similarity

Lab 5 · find streets that look alike.

An **embedding** is a list of numbers that describes an image, made so that similar images get similar lists. Comparing two lists with the **cosine similarity** (1 = same direction, 0 = unrelated) turns "find streets like this one" into simple arithmetic. Here we start with a small embedding: the 16 pixel shares and 6 perception scores of each image, 22 numbers.




```
Z  = StandardScaler().fit_transform(good[SEG + PER])     # 22 numbers per image, on the same scale
Zn = Z / np.linalg.norm(Z, axis=1, keepdims=True)         # unit length
sim = Zn @ Zn[q]                                          # cosine similarity of every image to
image q
top = np.argsort(-sim)[1:6]                               # the 5 most similar streets
```

```
images: 15881 | vector length: 22 | 2 PCA axes keep 21% of the variation
  axis1: low = sky, depressing, boring | high = safe, vegetation, beautiful
  axis2: low = terrain, wall, boring | high = car, building, lively
query image 62346ec0 (green 0.89, scene residential_neighborhood): 5 most similar images
  sim 0.974 |    29 m away | green 0.80 | scene forest_road
  sim 0.972 |  2218 m away | green 0.72 | scene residential_neighborhood
  sim 0.970 |    83 m away | green 0.77 | scene forest_road
  sim 0.966 |   908 m away | green 0.83 | scene residential_neighborhood
  sim 0.959 |   107 m away | green 0.84 | scene residential_neighborhood
  random pairs   0-0.1 km apart: mean similarity 0.15 (n=2823)
  random pairs 0.1-0.3 km apart: mean similarity 0.04 (n=10115)
  random pairs 0.3-1   km apart: mean similarity 0.01 (n=60677)
  random pairs   1-3   km apart: mean similarity -0.00 (n=126300)
```

The streets most similar to the greenest image are green residential and forest roads, some next door, some 2 km away. Similarity falls with distance (0.15 within 100 m, about 0 beyond 1 km), yet similar places can be far apart: that is what makes similarity search useful.

On your laptop. CLIP (Radford et al., 2021) turns each image into 512 learned numbers and puts words in the same space, so you can search images with a sentence: `embed_images.py`.

```
model, _, prep = open_clip.create_model_and_transforms('ViT-B-32', pretrained='openai')
E = model.encode_image(batch_of_images)                   # 512 numbers per image
t = model.encode_text(tok(['a shaded street with large trees']))
best = np.argsort(-(E @ t[0]))[:3]                        # search the city with words
```

7 Reviews Meet Streets

Lab 6 · joining what people see with what people write.

The reviews guide of this series built a comfort lens from 40,000 Singapore reviews. Each reviewed place has coordinates, so we can find the street images within 200 m of it and ask: do places on greener streets read differently?

```
tree = BallTree(np.radians(good[['lat', 'lon']].values), metric='haversine')
idx = tree.query_radius(np.radians(places[['lat', 'lon']].values), r=0.2 / 6371.0)   # 200 m
places['green_view_index'] = [good.green_view_index.values[i].mean() if len(i) else np.nan for i
in idx]
```

```
places with at least 3 good street images within 200 m: 43 of 5407
                  places  green  reviews  heat %  quiet %  noise %  atmosphere
green_band
least green third    15.0   0.31     97.0    1.03     4.12     3.09        4.09
middle third         14.0   0.41    115.0    0.87     4.35     0.87        4.54
greenest third       14.0   0.48    102.0    1.96     3.92     3.92        4.46
share of shopping malls: least green third 13% | greenest third 0%
```

Read this honestly. Only 43 of 5,407 reviewed places fall inside the image patch, with about 100 reviews per band. The differences in the table are within what chance would produce with so few reviews. The method works; the data are too thin to answer the question. This is exactly what Part B is for: download images around many more places, then repeat Lab 6.

CHALLENGE A · POWER

How many places per band would you need to detect a difference of 1 percentage point in heat complaints? Simulate it: shuffle the green bands 1,000 times and look at the spread of the differences.

ZenSVI on Your Laptop

Your own images, around your own places.

Part B runs the full pipeline. It downloads open Mapillary images within 100 m of 30 cafes from the reviews sample, segments them, scores them for perception, and joins everything back to the reviews. The full script is `zensvi_lab.py`.

BEFORE CLASS

1. `pip install zensvi` (installs PyTorch; a laptop CPU is enough for a few hundred images).
2. Create a free token at mapillary.com/dashboard/developers and save it yourself in `~/.mapillary_token`. Never paste a token into slides, chats or shared code.
3. Put `sg_places.csv` and `sg_reviews.csv` next to the script.

Step 1 · download. ZenSVI finds every image within the buffer of each point and saves the image list with its coordinates and capture date.

```
from zensvi.download import MLYDownloader
TOKEN = (Path.home() / '.mapillary_token').read_text().strip()   # your own free token
mly = MLYDownloader(mly_api_key=TOKEN)
mly.download_svi('svi_sg', input_csv_file='pick.csv', buffer=100, resolution=1024)
#  pick.csv has columns latitude, longitude  ->  svi_sg/mly_pids.csv, svi_sg/mly_svi/batch_1/...
```

Steps 2 and 3 · segment and perceive. The segmenter uses a model trained on Mapillary Vistas classes; the perception classifier uses one model per Place Pulse question.

```
from zensvi.cv import Segmenter, ClassifierPerception
seg = Segmenter(dataset='mapillary', task='semantic')
seg.segment('svi_sg/mly_svi/batch_1', dir_image_output='svi_sg/seg_images',
            dir_summary_output='svi_sg/seg_summary', csv_format='wide', save_format='csv')
clf = ClassifierPerception(perception_study='more beautiful')     # also: safer, livelier, ...
clf.classify('svi_sg/mly_svi/batch_1', dir_summary_output='svi_sg/beautiful', save_format='csv')
```

Step 4 · join. For each cafe, average the green view and perception of its nearby images, then join the review comfort scores. The script prints one row per cafe and saves `places_street_and_reviews.csv`.

CHALLENGE B · YOUR CITY

Change `pick.csv` to 30 places in Harbin or your home city. Compare their green view index with the Singapore patch. What would you need to compare them fairly (season, year, platform)?

Pitfalls, Ethics and Glossary

Read this before you publish a street indicator.

Pitfalls checklist

- **Coverage.** Open images cluster where volunteers drive. Map coverage before analysis.
- **Time.** Mixed years and seasons. In cold cities, winter trees have no leaves.
- **Platform.** Car, bike and foot views see different streets.
- **Model labels.** Count impossible labels (snow in Singapore).
- **Spatial leakage.** Use spatial blocks for cross-validation.
- **Perception is borrowed.** Place Pulse votes came mostly from online volunteers; “beautiful” may not transfer across cultures.
- **Seen is not felt.** Green view is not shade or temperature.

Ethics

- Faces and number plates are blurred on open platforms; do not un-blur or re-identify.
- Keep the licence (CC BY-SA) and credit the photographers’ platforms.

Glossary

Street view imagery (SVI)
Photos taken from streets at eye level.

Mapillary, KartaView
Open, crowd-sourced street image platforms.

Semantic segmentation
Giving every pixel a class (tree, road…).

Pixel share
Fraction of the image in one class.

Green view index
Share of vegetation as seen from the street.

Sky view index
Share of sky in the image.

Place Pulse
Crowd-sourced pairwise votes on street images.

Perception score
A model’s estimate of those votes, 0–10.

Places365
A scene classifier with 365 scene types.

Spatial block
A group of nearby samples kept together in cross-validation.

Embedding
A vector that describes an image; similar images get similar vectors (cosine similarity).

ZenSVI
Python package to download and analyse SVI.

One-Page Cheat Sheet

Everything in this guide on one page.

INSTALL

pip install zensvi

DOWNLOAD

MLYDownloader(mly_api_key=TOKEN).download_svi(out, lat=..., lon=..., buffer=100)

SEGMENT

Segmenter(dataset='mapillary').segment(dir, dir_summary_output=...)

PERCEIVE

ClassifierPerception('more beautiful').classify(dir, dir_summary_output=...)

GREEN VIEW

GVI = (vegetation + terrain) pixels / all pixels

SKY VIEW

SVI_{sky} = sky pixels / all pixels

FILTER FIRST

Quality, glare, view direction, platform, year. Report your choices.

VALIDATE

Spatial blocks (GroupKFold on 1–2 km cells).

JOIN TO PLACES

BallTree(haversine).query_radius(points, r = metres / 6,371,000)

LAB RESULTS (SINGAPORE PATCH)

GVI 0.44 · green ↔ beautiful 0.49 · R^2 beautiful 0.33 · 480 “snowy” images.

DATA

Global Streetscapes on Hugging Face: NUS-UAL/global-streetscapes (CC BY-SA 4.0).

PAIR WITH

Reviews (how it feels) · sensors (how hot it is) · 3D city (why).

Further Reading

The papers and tools behind each idea.

1. Ito, K., Zhu, Y., Abdelrahman, M., Liang, X., Fan, Z., Hou, Y., Zhao, T., Ma, R., Fujiwara, K., Ouyang, J., Quintana, M., & Biljecki, F. (2025). ZenSVI: an open-source software for the integrated acquisition, processing and analysis of street view imagery towards scalable urban science. *Computers, Environment and Urban Systems*, 102283.
2. Hou, Y., Quintana, M., Khomiakov, M., Yap, W., Ouyang, J., Ito, K., Wang, Z., Zhao, T., & Biljecki, F. (2024). Global Streetscapes: a comprehensive dataset of 10 million street-level images across 688 cities for urban science and analytics. *ISPRS Journal of Photogrammetry and Remote Sensing*. doi:10.1016/j.isprsjprs.2024.06.023
3. Biljecki, F., & Ito, K. (2021). Street view imagery in urban analytics and GIS: a review. *Landscape and Urban Planning*, 215, 104217.
4. Li, X., Zhang, C., Li, W., Ricard, R., Meng, Q., & Zhang, W. (2015). Assessing street-level urban greenery using Google Street View and a modified green view index. *Urban Forestry & Urban Greening*, 14(3), 675–685.
5. Dubey, A., Naik, N., Parikh, D., Raskar, R., & Hidalgo, C. A. (2016). Deep learning the city: quantifying urban perception at a global scale. *ECCV*.
6. Naik, N., Philipoom, J., Raskar, R., & Hidalgo, C. (2014). Streetscore: predicting the perceived safety of one million streetscapes. *CVPR Workshops*.
7. Neuhold, G., Ollmann, T., Rota Bulò, S., & Kontschieder, P. (2017). The Mapillary Vistas dataset for semantic understanding of street scenes. *ICCV*.
8. Radford, A., et al. (2021). Learning transferable visual models from natural language supervision (CLIP). *ICML*.
9. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2018). Places: a 10 million image database for scene recognition. *IEEE TPAMI*, 40(6), 1452–1464.
10. Quintana, M., et al. (2025). Global urban visual perception varies across demographics and personalities. *Nature Cities*.

Files · `svi_lab.py` · `svi_lab.ipynb` · `sg_svi.csv` (Global Streetscapes, CC BY-SA 4.0) · `zensvi_lab.py`, `embed_images.py` (Part B)

Part of the Fundamentals 101 series · mahmoud.work