Building a visual similarity search from scratch

SLO AI Lightning Talks #2
December 11, 2025

Robert Sedovšek
https://galjot.si

To break the ice...

Career so far

¯_(ツ)_/¯ (*now*)
Turtl (*before*)
Celtra (*way before*)

Normal hobbies

Running, XC ⛷️ and ⛰️ 🚵
Photography

PhD

Faculty of Natural Sciences and Engineering;
University of Ljubljana

Unusual trivia

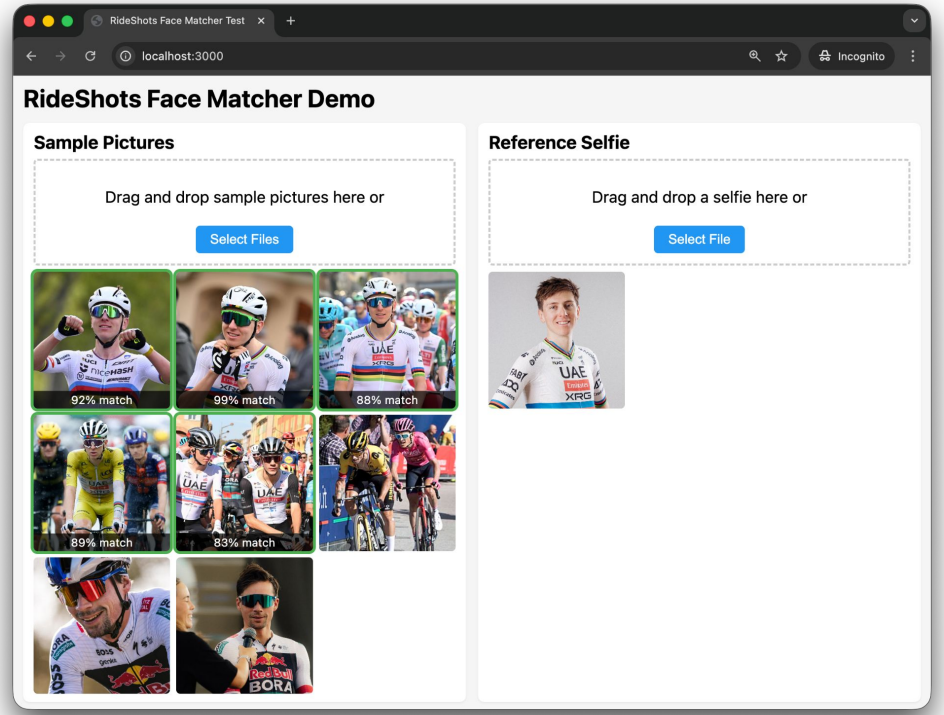
I ran a **100k** (5,500m+ elevation) race **twice**

Background & motivation

RideShots.cc (*passion project*)

Image search via **facial recognition:**

- ~~face-api.js~~
- ~~face_recognition~~
- ~~Insightface~~
- **AWS Rekognition**



Let’s build it!

— from scratch, with open-source tools —

On a **concrete case**: finding a specific item
(e.g., a dress) within a catalog of product images.

✓ What to expect?

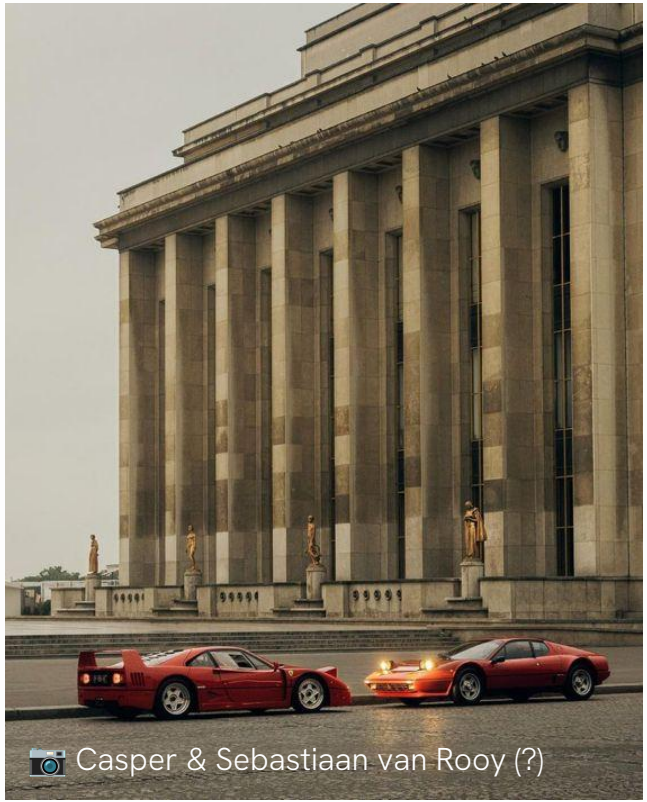
1. **Data collection**
2. **Object detection**
   locating the dress in the image
3. **Segmentation & background removal**
   isolating the dress itself
4. **Embedding**
   turning the image into numerical vectors
5. **Search**
   finding similar embeddings

✗ What this is not?

- Scalability/perf. concerns
- Production ready setup
- Alternatives:
  numerous other models I tried and tested, less successfully, and not used in the demo

Which two photos are more similar? (1/2)

Which two photos are more similar? (2/2)



📷 Casper & Sebastiaan van Rooy (?)

📷 unknown

📷 Calvin Courjon

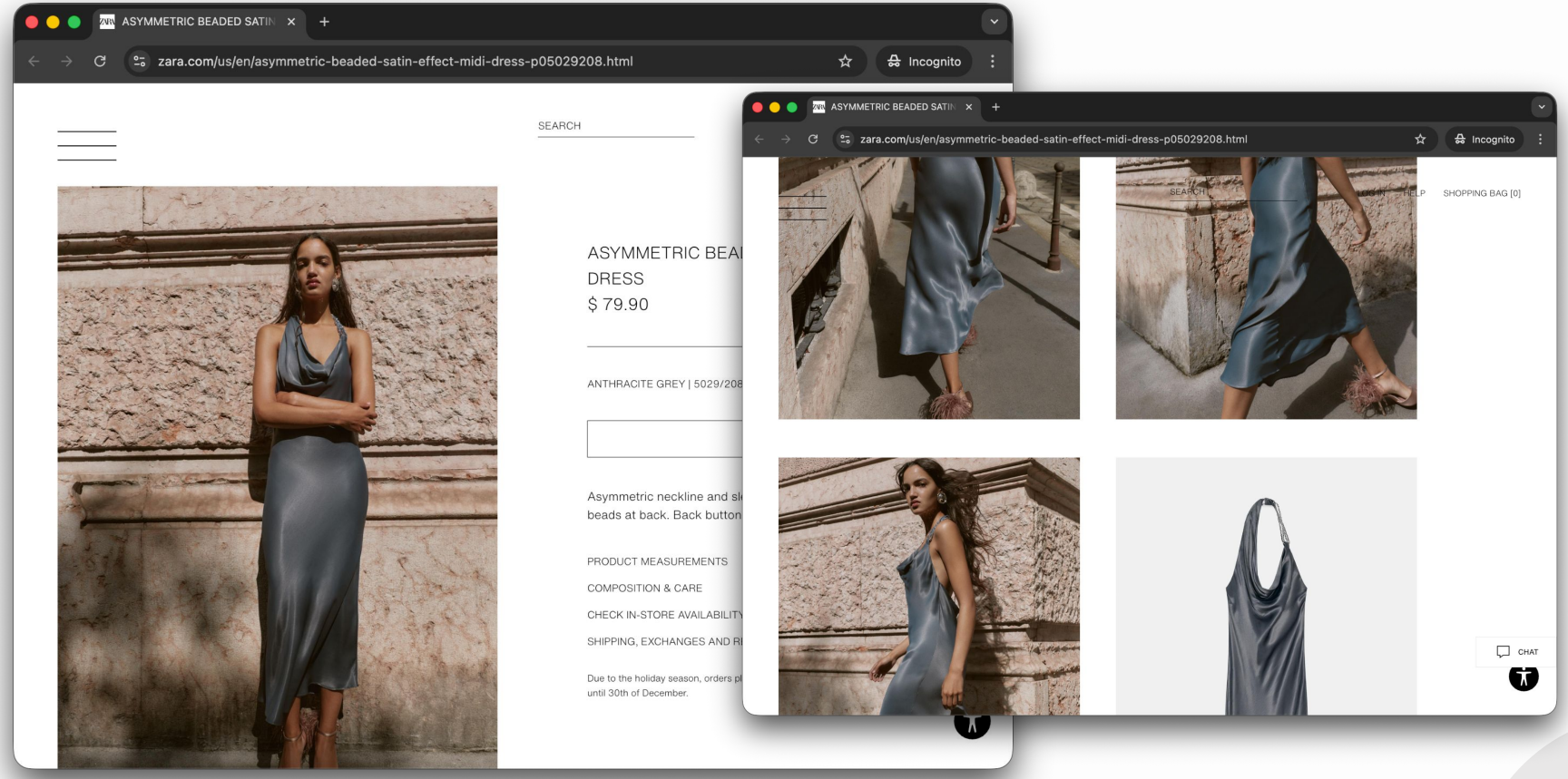
1. Data collection

Technologies

- **Puppeteer** — headless browser automation
- **Axios** — HTTP requests for image downloads
- **Node.js** — file system operations

Process

- Reads product URLs from a text file
- Navigates pages with Puppeteer
- Simulates scrolling to trigger lazy-loaded images
- Extracts image URLs via DOM selectors
- Downloads images to local storage


ASYMMETRIC BEADED SATIN
zara.com/us/en/asymmetric-beaded-satin-effect-midi-dress-p05029208.html
Incognito
SEARCH
ASYMMETRIC BEA
DRESS
$ 79.90
ANTHRACITE GREY | 5029/208
Asymmetric neckline and sl
beads at back. Back button
PRODUCT MEASUREMENTS
COMPOSITION & CARE
CHECK IN-STORE AVAILABILITY
SHIPPING, EXCHANGES AND R
Due to the holiday season, orders pl
until 30th of December.
ASYMMETRIC BEADED SATIN
zara.com/us/en/asymmetric-beaded-satin-effect-midi-dress-p05029208.html
Incognito
SEARCH
HELP
SHOPPING BAG [0]
CHAT

2. Object detection

Term

Object detection

Task

Find and label objects in an image

Why it matters?

Identify what’s in an image

Model: Florence-2 Object Detection (Microsoft)

A lightweight vision-language model open-sourced by Microsoft under the MIT license.

```python
# Object Detection: Finding the dress in the image

from transformers import AutoProcessor, AutoModelForCausalLM

# Load Florence-2 (processor for image/text encoding, model for inference)
processor = AutoProcessor.from_pretrained("microsoft/Florence-2-base")
model = AutoModelForCausalLM.from_pretrained("microsoft/Florence-2-base")

def detect_clothing(image):
    od_task = "<OD>" # Object Detection

    inputs = processor(text=od_task, images=image, return_tensors="pt")
    detections = model.generate(**inputs)
    parsed = processor.post_process_generation(
        detections, task=od_task, image_size=(image.width, image.height)
    )

    bbox  = parsed['<OD>']['bbox']
    label = parsed['<OD>']['label']

    return None if label not in ['dress', 'shirt', 'pants', 'jacket', ...]:

    return {'name': label, 'bbox': bbox}

```

3. Segmentation & background removal

Term

Segmentation

Task

Pixel-level object mask (isolating the dress itself)

Why it matters?

Enables background removal and object isolation

Model: SAM 2 – Segment Anything Model v2 (Meta)

Designed specifically to generate pixel-accurate masks for any object in an image.

Segmentation & Background Removal: Isolating the dress

```python
from transformers import Sam2Model, Sam2Processor

sam2_processor = Sam2Processor.from_pretrained("facebook/sam2-hiera-large")
sam2_model = Sam2Model.from_pretrained("facebook/sam2-hiera-large")

def segment_clothing(image, bbox):
    inputs = sam2_processor(
        images=image,
        input_boxes=[[[bbox[0], bbox[1], bbox[2], bbox[3]]]],
        return_tensors="pt"
    )
    outputs = sam2_model(**inputs, multimask_output=False)
    processed_masks = sam2_processor.post_process_masks(
        outputs.pred_masks.cpu(),
        inputs["original_sizes"]
    )
    return processed_masks[0][0][0].numpy() > 0  # Pick best mask

def remove_background(img, mask):
    original_rgb = np.array(img)
    bg_removed = np.zeros((img.height, img.width, 4)) # Transparent image

    for y in range(img.height):
        for x in range(img.width):
            if mask[y, x]: # If mask is True, keep pixel
                bg_removed[y, x] = [*original_rgb[y, x], 255]

    return PILImage.fromarray(bg_removed, 'RGBA')

```

4. Embedding

Term

Embedding

Task

Convert a photo into a
numerical vector representing its features.

Why it matters?

Enables comparison between vectors
(*similar images*)

Model: FashionCLIP

(patrickjohncyh/fashion-clip)

A vision-language model based on CLIP (**OpenAI**), fine-tuned on fashion data.

Embedding: Turning the image into numerical vectors

```python
from transformers import CLIPModel, CLIPProcessor

model = CLIPModel.from_pretrained("patrickjohncyh/fashion-clip")
processor = CLIPProcessor.from_pretrained("patrickjohncyh/fashion-clip")

def generate_embedding(garment_image):
    inputs = processor(images=garment_image, return_tensors="pt")
    embedding = model.get_image_features(**inputs)

    return embedding[0].numpy() # Returns: 512-dimensional vector
```

5. Search

Term

Search

Task

Find similar images by comparing embedding vectors

Why it matters?

Enables fast retrieval of visually similar products

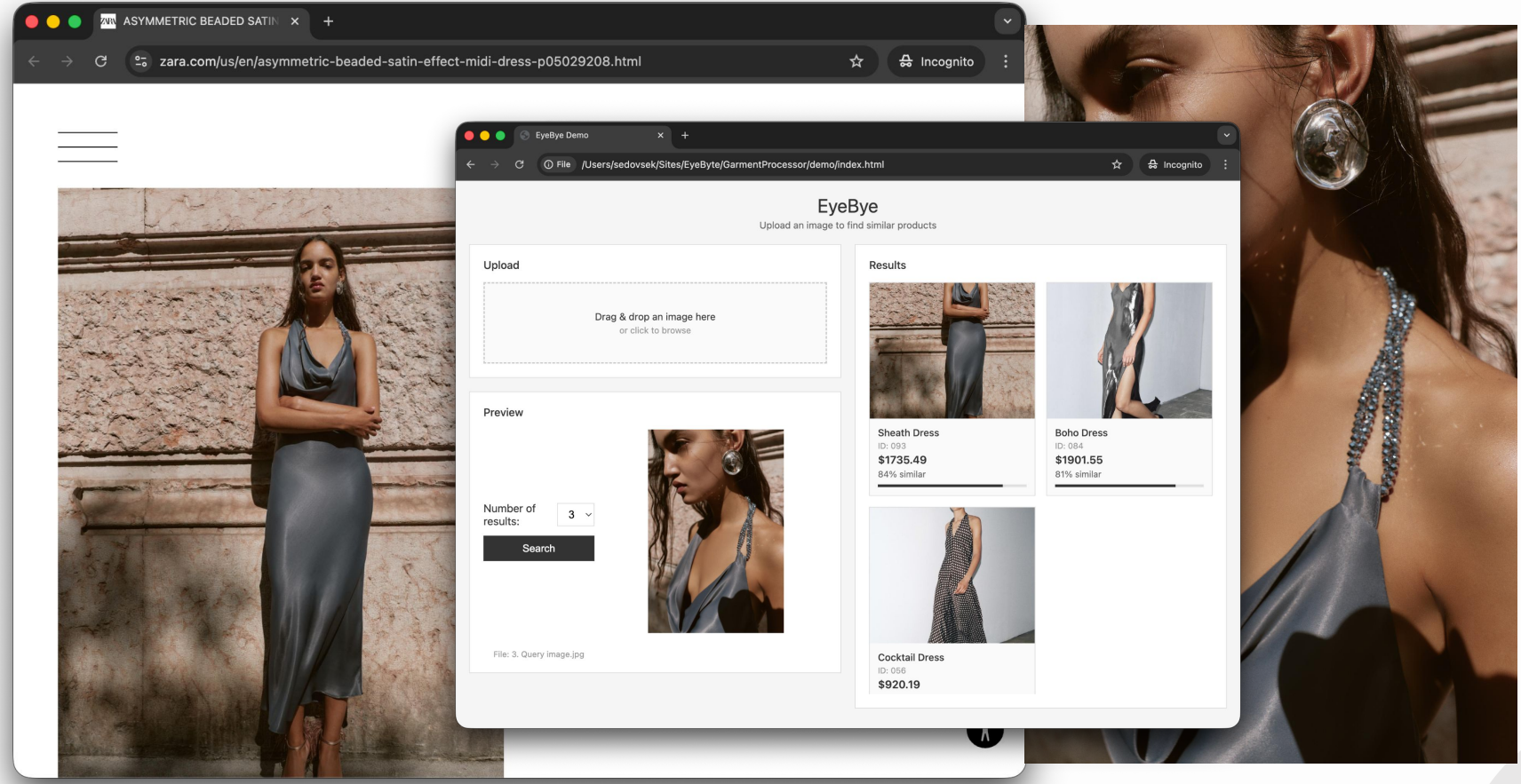
Search

🤗 sedovsek/garment-similarity-search

For each query image, compute **cosine similarity** against all product embeddings.

Search: Finding similar embeddings

```python
import numpy as np

class SimpleVectorIndex:
    def __init__(self):
        self.embeddings, self.product_ids = []

    def insert(self, product_id, embedding):
        self.embeddings.append(embedding)
        self.product_ids.append(product_id)

    def search(self, query_embedding, k=5):
        # CLIP embeddings are already normalized, so we can compute
        # cosine similarity directly via dot product
        similarities = np.dot(self.embeddings, query_embedding)

        # Get top-k most similar products
        top_k_indices = np.argsort(similarities)[::-1][:k]

        results = []
        for idx in top_k_indices:
            results.append({
                'id': self.product_ids[idx],
                'similarity': float(similarities[idx])
            })

        return results

```


ASYMMETRIC BEADED SATIN
zara.com/us/en/asymmetric-beaded-satin-effect-midi-dress-p05029208.html
Incognito
EyeBye Demo
File
/Users/sedovsek/Sites/EyeByte/GarmentProcessor/demo/index.html
Incognito
EyeBye
Upload an image to find similar products
Upload
Drag & drop an image here
or click to browse
Preview
Number of results:
3
Search
File: 3. Query image.jpg
Results
Sheath Dress
ID: 093
$1735.49
84% similar
Boho Dress
ID: 084
$1901.55
81% similar
Cocktail Dress
ID: 056
$920.19

Pipeline

```python
# Visual Similarity Search Pipeline

vector_index = SimpleVectorIndex()

# OFFLINE: Build product index
for product in load_products("data/products.json"):
    for image in product["images"]:
        embedding = process_image(Image.open(image))
        vector_index.insert(product["id"], embedding)

# CLIENT: Search for similar products
query_embedding = process_image(Image.open("upload.jpg"))
similar_products = vector_index.search(query_embedding, k=5)

def process_image(image):
    """Process: detect → segment → remove background → embed"""
    detection = detect_clothing(image)
    mask = segment_clothing(image, detection["bbox"])
    garment = remove_background(image, mask)
    return generate_embedding(garment)

```

Q&A

Thank you for your attention!
You could have been scrolling through your phone, but you chose not to –
I appreciate it!

@sedovsek on X · Instagram · Threads · Strava
https://galjot.si