UNIVERSITÀ
DI TRENTO

UNIVESITÀ DI TRENTO
REPARTO DISI
ARTIFICIAL INTELLIGENCE SYSTEM

CERPELLONI DAVIDE

SIGNAL IMAGE AND VIDEO PROCESSING

STITCHING IMAGES E BARCODE DETECTION

Trento-TN
2024

1 Introduction

The idea of this project come from a necessity of an agency to have a scanner of product that elaborate it and then recognize and check various label.

At the moment they have to manually take different photos of the product from different point of view, save all of them, checking the different labels in the different images of the product and write it down manually.

To automate this process the project at his best comprehend a setting in which I place an object in a defined position, a camera go round it and make some photos of the product from different point of view. Then these photos are stitched together to create a unique image of the product's label and a text recognition is performed to get various information about the product as the name, nutritional values, expiration date and so on.

In this project I concern with the part of stitching process, starting already from the product label images, stitching them together and then looking for the barcode. Once having it I perform an api call to get the necessary information about the product. For this project I will use Python, the most common language to perform task of image processing like this.

2 Main Components

2.1 SIFT

SIFT stands for Scale-Invariant Feature Transform, and it's a powerful algorithm used in image processing for detecting and describing keypoints (distinctive features) in images. These keypoints are especially useful for tasks like:

- Object recognition: By comparing the keypoints of an unknown object to a database of known objects, we can identify what the object is.
- Image matching: We can find corresponding points between two images, even if they're taken from different angles, distances, or lighting conditions. This is crucial for tasks like stitching panoramas or tracking objects in videos.
- 3D reconstruction: By matching keypoints across multiple images of a scene, we can create a 3D model of the scene.

2.2 How SIFT works

1. Keypoint detection: SIFT uses a multi-scale approach to find interesting points in the image that are stable under changes in scale and rotation. It does this by creating multiple versions of the image with different levels of blur and then looking for points that are stable across these different scales.
2. Descriptor extraction: Once the keypoints are found, SIFT extracts a descriptor for each one. This descriptor is a 128-dimensional vector that captures the local properties of the image around the keypoint. These descriptors are designed to be robust to changes in illumination, noise, and perspective.
3. Matching: Finally, SIFT can be used to match keypoints between two images. This is done by comparing the descriptors of the keypoints in one image to the descriptors of the keypoints in the other image and finding the pairs of descriptors that are most similar.

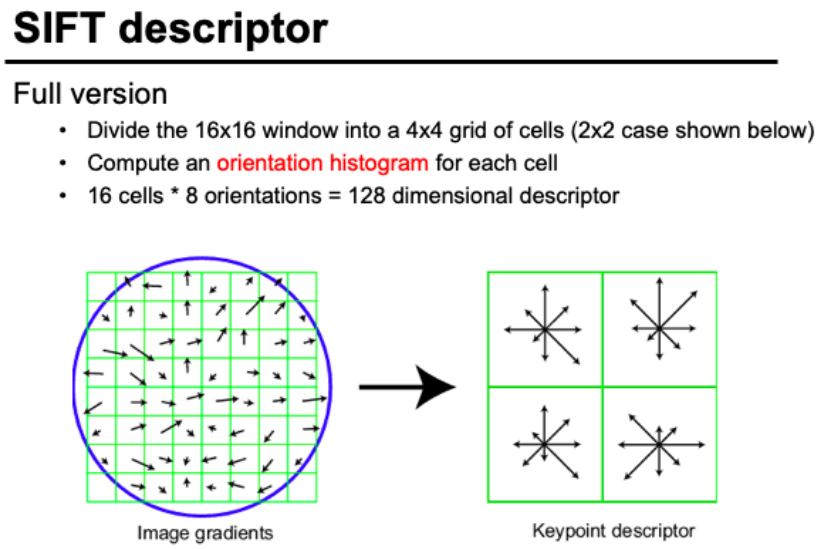

Figura 1. *Descriptor extraction*

2.2.1 Advantages of SIFT:

- Scale and rotation invariance: As the name suggests, SIFT is very good at finding keypoints that are stable under changes in scale and rotation. This makes it very useful for tasks like object recognition and image matching, where images may be taken from different angles or distances.
- Distinctive features: SIFT descriptors are very distinctive, which means that they are unlikely to be accidentally matched to the wrong keypoint. This is important for tasks like object recognition, where accurate matching is essential.

2.2.2 Disadvantages of SIFT:

- Computational cost: SIFT can be computationally expensive, especially on large images. This can be a limitation for real-time applications.

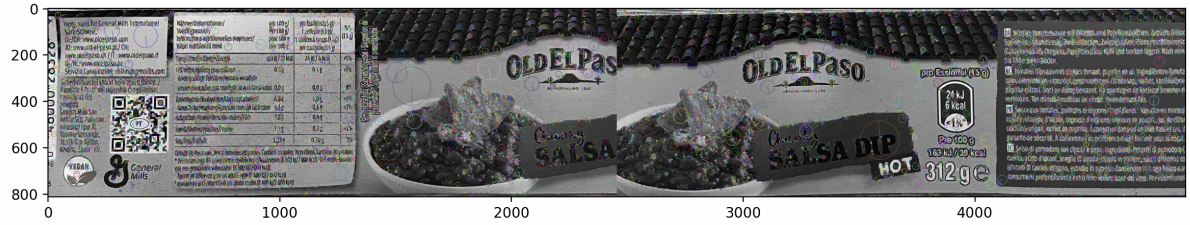

Figura 2. *Matching between my two images*

2.3 Brute Force Matcher

The next step after applying SIFT algorithm, we apply the Brute Force algorithm to obtain all points that our two images have in common.

Brute-Force matcher is simple. It takes the descriptor of one feature in first set and is matched with all other features in second set using some distance calculation. And the closest one is returned.

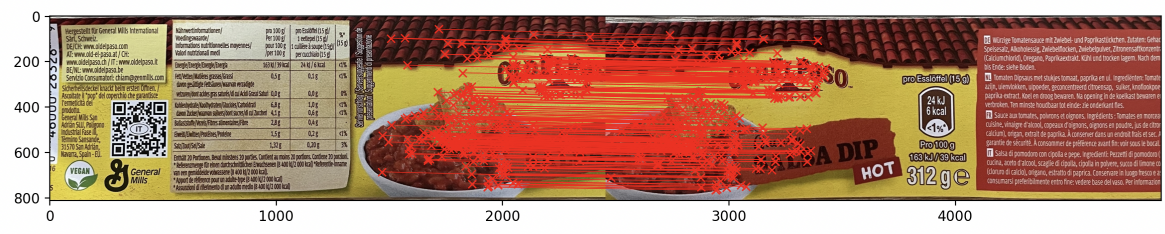

Figura 3. *Brute force between my two images*

2.4 Homography

Homography, also known as planar homography, is a powerful concept in image processing that describes the geometric relationship between two images of the same planar surface taken from different viewpoints. Essentially, it's a mathematical transformation that maps corresponding points between these images.

Think of it this way: Imagine taking a picture of a flat piece of paper from different angles. Even though the perspective changes, the actual geometric relationships between points on the paper remain constant. Homography captures this underlying relationship mathematically.

Key characteristics:

- Applies to planar surfaces: It's most accurate for objects that can be approximated as flat, like documents, billboards, or ground planes.
- Represented by a 3x3 matrix: This matrix encodes the scaling, rotation, and translation between the two images.
- Used for various tasks: Homography has numerous applications in image processing, including:
  - Image rectification: Correcting distortions caused by perspective changes.
  - Image registration: Aligning overlapping images from different viewpoints.
  - Object tracking: Following an object across multiple frames by matching corresponding points.
  - Panorama stitching: Creating seamless panoramas from multiple images.
  - Augmented reality: Superimposing virtual objects onto real-world scenes with accurate perspective.

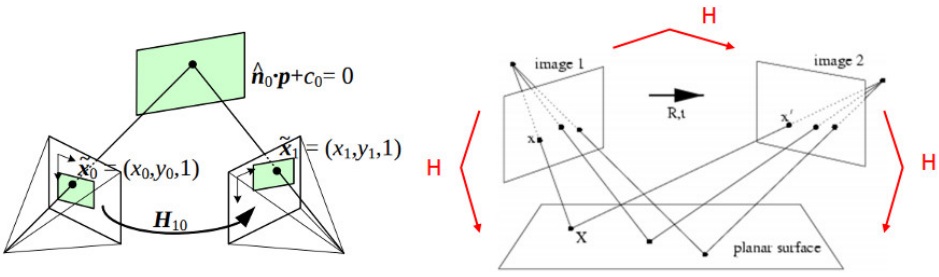

Figura 4. *Homography*

2.5 Ransac

RANSAC stands for RANdom SAmple Consensus. It's a powerful technique used to estimate mathematical models (like lines, planes, or homographies) from a set of data points that might contain outliers. In image processing, these outliers can be caused by:

- Noise: Random fluctuations in image data.
- Mismatches: Incorrectly matched features between images.
- Occlusions: Objects being partially hidden in the image.

2.5.1 How does it work?

If you want to fit a line to a set of points, but some of those points are far away from the true line due to outliers. Here's how RANSAC tackles this:

1. Randomly sample: It repeatedly selects a small subset of points (usually the minimum number needed to estimate the model).
2. Fit the model: Using the chosen points, it estimates the model parameters (e.g., the slope and intercept of the line).
3. Count inliers: It counts how many other data points fall within a certain distance (tolerance) of the estimated model.
4. Repeat and compare: It repeats steps 1-3 many times, each time keeping track of the model with the most inliers.
5. Choose the best: Finally, it selects the model with the highest number of inliers as the most likely representation of the actual data, excluding the outliers.

2.5.2 Benefits of using RANSAC:

- Robust to outliers: It's less sensitive to the presence of outliers compared to traditional fitting methods.
- Iterative approach: It can handle situations where the initial random sample might not be the best, as it keeps iterating and improving.

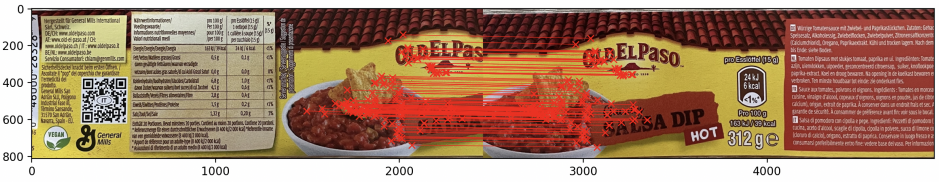

Figura 5. *Inliers after applying RANSAC*

2.6 Stitching Images

The last step, after applying RANSAC, to get ouliers and homography matrix, is stitching the two subject images.

2.6.1 How does it work

The basic steps involved in image stitching are:

- Feature detection and matching: Identifying and matching distinctive features (like corners, edges, or keypoints) between overlapping regions of the images.

- Image alignment: Estimating the geometric transformation (e.g., rotation, translation) required to align each image based on the matched features.
- Blending and seam removal: Seamlessly merging the aligned images and removing any visible seams where they join, creating a natural-looking panorama.

Figura 6. *This is the final result*

2.7 Detect barcode and retrieve labels

Now to detect the barcode I use a library in python that using OpenCV, PIL and numpy can detect and decode the number of the product. Then I perform an api call to retrieve all information that i need about my product like name and nutritional values.

3 Future Implementation

The stitching process represented above is only the body of the whole project. Then we have others tasks to complete the project, indeed we have to:

- Automated the photos acquisition: the product will be placed in a defined position and then a camera should rotato around it to take the photos from different point of view.
- Isolate the product label: once we have the product photos we have to identify and isolate only the label of the product on each photo.
- After stitching process is executed, we have to implement our custom Ocr to identify the barcode and to detect all text printed on product labels, identifying various information that now we retrieve through Api call.