Report

Submitted by: Saloni
Date: 21-05-2021

Data processing techniques

- Read image
- Resize image
- Remove noise(Denoise)
- Segmentation (The segmentation step is only useful for segmentation problems, if the AI -Computer Vision problem does not include segmentation, we should skip this step.)
- Morphology(smoothing edges)

The segmentation step is only useful for segmentation problems, if your AI -Computer Vision problem does not include segmentation, just skip this step.

Keypoint

- Keypoint detection involves simultaneously detecting people and localizing their keypoints.
- Keypoints are the same thing as interest points. They are spatial locations, or points in the image that define what is interesting or what stand out in the image.
- They are invariant to image rotation, shrinkage, translation, distortion, and so on.

Feature Extraction: Traditional and Deep Learning Techniques

Feature Extraction is an important technique in Computer Vision widely used for tasks like:

- Object recognition
- Image alignment and stitching (to create a panorama)
- 3D stereo reconstruction
- Navigation for robots/self-driving cars, etc.

What are features?

Features are parts or patterns of an object in an image that help to identify it.

For example - a square has 4 corners and 4 edges, they can be called features of the square, and they help us humans identify it's a square. Features include properties like corners, edges, regions of interest points, ridges, etc.

Glimpse of Traditional feature detection techniques

Traditional Computer Vision techniques for feature detection include:

- **Harris Corner Detection**: Uses a Gaussian window function to detect corners.
- **Shi-Tomasi Corner Detector:** The authors modified the scoring function used in Harris Corner Detection to achieve a better corner detection technique.
- **Scale-Invariant Feature Transform (SIFT):** This technique is scale invariant unlike the previous two.
- **Speeded-Up Robust Features (SURF):** This is a faster version of SIFT as the name says.
- **Features from Accelerated Segment Test (FAST):** This is a much faster corner detection technique compared to SURF.

- **Binary Robust Independent Elementary Features (BRIEF):** This is only a feature descriptor that can be used with any other feature detector. This technique reduces the memory usage by converting descriptors in floating point numbers to binary strings.
- **Oriented FAST and Rotated BRIEF (ORB):** SIFT and SURF is patented and this algorithm from OpenCV labs is a free alternative to them that uses FAST key point detector and BRIEF descriptor.

Glimpse of AI based feature extraction techniques

Traditional feature extractors can be replaced by a convolutional neural network(CNN), since CNN's have a strong ability to extract complex features that express the image in much more detail, learn the task specific features and are much more efficient.

- **SuperPoint**
  It suggests a fully convolutional neural network that computes SIFT like interest point locations and descriptors in a single forward pass. It uses an VGG-style encode for extracting features and then two decoders, one for point detection and the other for point description.
- **D2-Net**
  It suggests a single convolutional neural network that is both a dense feature descriptor and a feature detector.
- **LF-Net**
  It suggest using a sparse-matching deep architecture and use an end-to-end training approach on image pairs having relative pose and depth maps.
- **Image Feature Matching Based on Deep Learning**
  It is a deep Convolutional neural network (CNN) model, which attention on image patch, in image feature points matching.

Feature Matching

Features matching or generally image matching, a part of many computer vision applications such as image registration, camera calibration and object recognition, is the task of establishing correspondences between two images of the same scene/object. A common approach to image matching consists of detecting a set of interest points each associated with image descriptors from image data. Once the features and their descriptors have been extracted from two or more images, the next step is to establish some preliminary feature matches between these images.

Feature Matching Techniques

1. **Brute-Force Matcher**

- Brute Force Matcher is used for matching the features of the first image with another image.
- It takes one descriptor of first image and matches to all the descriptors of the second image and then it goes to the second descriptor of first image and matches to all the descriptor of the second image and so on.

2. **FLANN (Fast Library for Approximate Nearest Neighbours) Matcher**

- FLANN stands for Fast Library for Approximate Nearest Neighbours. It contains a collection of algorithms optimized for fast nearest neighbour search in large datasets and for high dimensional features.
- It works faster than BFMatcher for large datasets.

- For FLANN based matcher, we need to pass two dictionaries which specifies the algorithm to be used, its related parameters etc.
- First one is IndexParams. Second dictionary is the SearchParams.

Dataset

Here we are using Industrial Estate RGB + thermal images and extracting some basic Features from it.

```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        from skimage.io import imread,imshow
        image=imread('EP-00-00012_0119_0001.JPG',as_gray=True)
        imshow(image)
```

```
Out[2]: <matplotlib.image.AxesImage at 0x1e3c64802b0>
```


```
[4]: #Grayscale Pixel Values as Features
     image=imread('EP-00-00012_0119_0001.JPG',as_gray=True)
     image.shape,imshow(image)
```

```
[4]: ((3648, 5472), <matplotlib.image.AxesImage at 0x20d1db49b70>)
```


```
In [8]: #Mean Pixel Value of Channels
        image=imread('EP-00-00012_0119_0001.JPG')
        feature_matrix=np.zeros((3648,5472))
        feature_matrix.shape
        for i in range(0,image.shape[0]):
            for j in range(0,image.shape[1]):
                feature_matrix[i][j]=((int(image[i,j,0])+int(image[i,j,1])+int(image[i,j,2]))/3)
                features=np.reshape(feature_matrix,(3648*5472))
        features.shape
```

```
Out[8]: (19961856,)
```

```
In [11]: #Extracting Edge Features
         from skimage.filters import prewitt_h,prewitt_v
         image = imread('EP-00-00012_0119_0001.JPG',as_gray=True)

         #calculating horizontal edges using prewitt kernel
         edges_prewitt_horizontal = prewitt_h(image)

         #calculating vertical edges using prewitt kernel
         edges_prewitt_vertical = prewitt_v(image)

         imshow(edges_prewitt_vertical, cmap='gray')
```

```
Out[11]: <matplotlib.image.AxesImage at 0x20d1ebc92b0>
```