Filippo Ficarra: Lab 7 - Structure from Motion and Line Fitting

fficarra@student.ethz.ch, 22–938–062.

21/12/2023 - 21:26h

1 Structure from Motion

1.1 Implementation

The principal functions that had to be implemented were:

- Essential matrix estimation
- Point triangulation
- Map extension

1.1.1 Essential matrix estimation

1.1.2 Point triangulation

1.1.3 Map extension

1.2 Results

The following imges show the result for the algorithm using all the images. Since the last 2 images contain some points far in the background, the cloud of points shows also them.

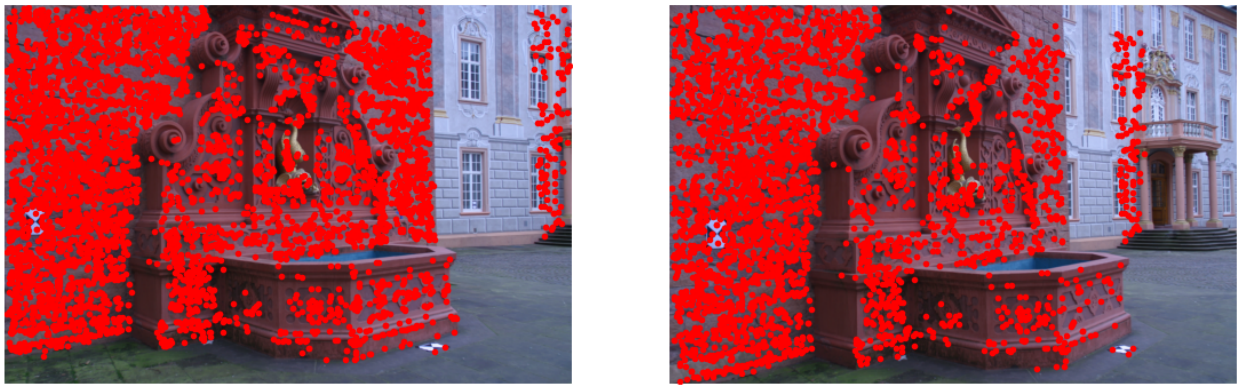

Figure 1: Images 8 and 9 with the points in the background.

If we remove the last two images we can clearly see the reconstruction of the fountain as shown below. The construction is much more visible, there are less points to be considered when computing and plotting.

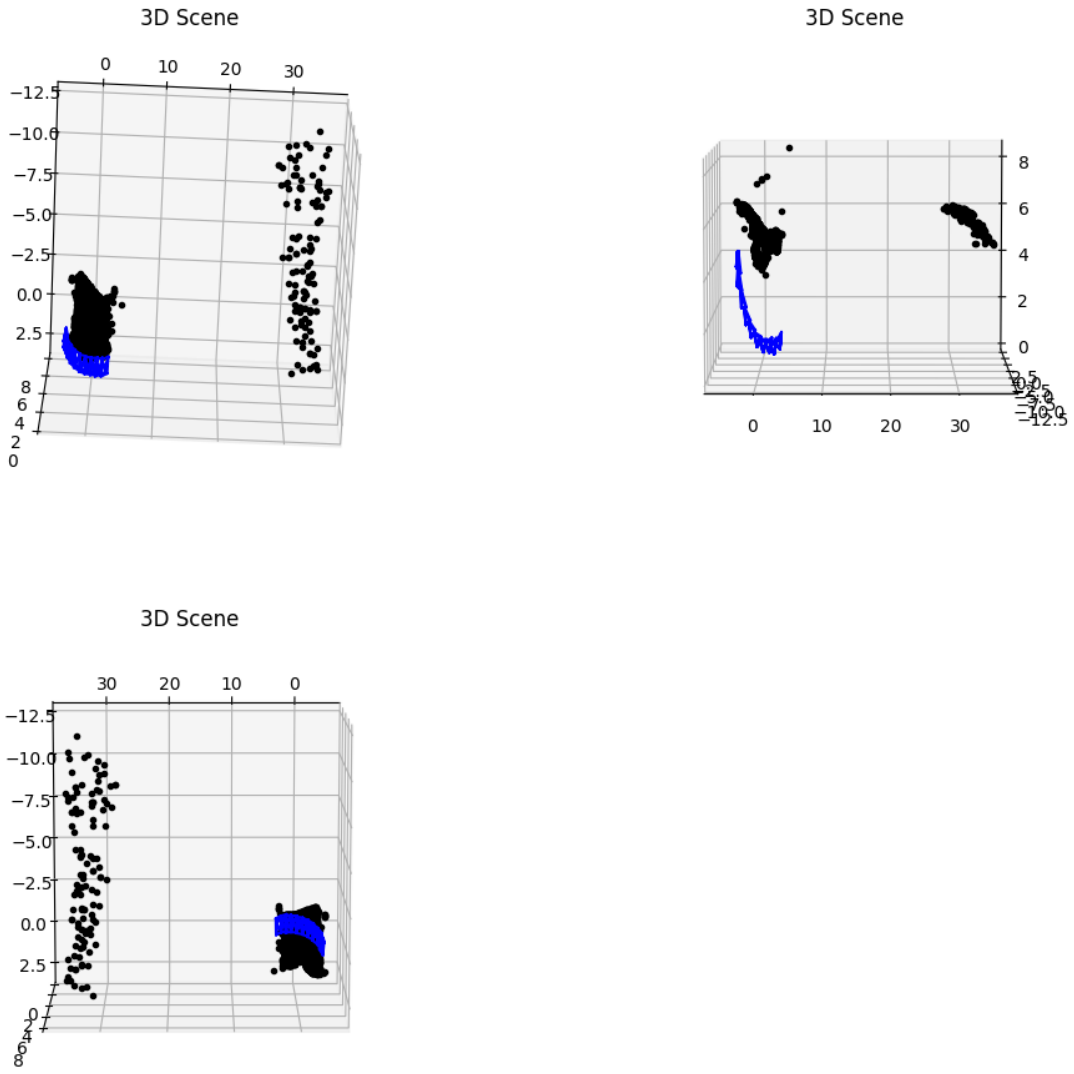

Figure 2: Results for all the images. From top left to bottom right: top-right frontal , top-frontal, back views.

2 Model Fitting

2.1 Implementation

The main function of the algorithm is shown below. The algorithm basically iterate all over the number of iterations, samples a random sample of points and compute the least squares. The least squares coefficient are then passed to a function to compute the number of outliers and their indices.

Listing 1: RANSAC

```
1  def ransac(x,y,iter,n_samples,thres_dist,num_subset):
2    k_ransac = None
3    b_ransac = None
4    inlier_mask = None
5    best_inliers = 0
6
7    for _ in range(iter):
```

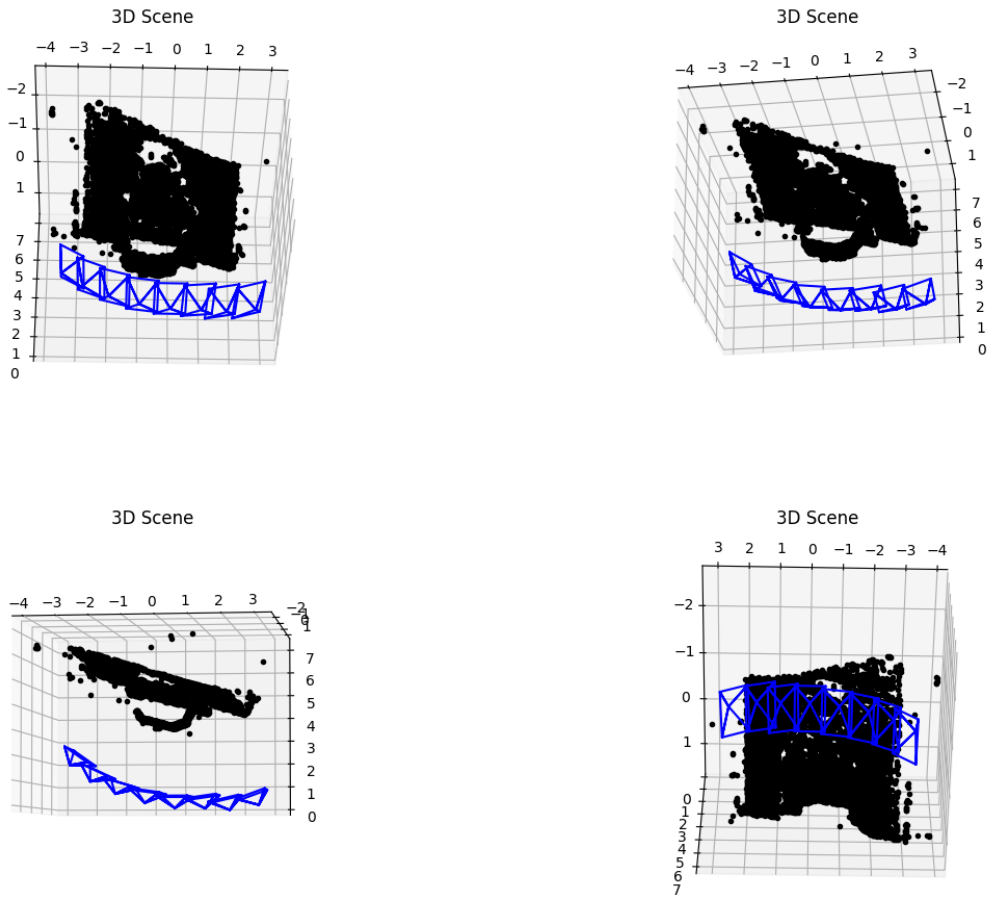

Figure 3: Results for images 0 to 7. From top left to bottom right: frontal, top-frontal, top, back views.

```
    indices = random.sample(range(n_samples), num_subset)

    k, b = least_square(x[indices], y[indices])

    num, mask = num_inlier(x, y, k, b, n_samples, thres_dist)

    if num > best_inliers:
      best_inliers = num
      k_ransac = k
      b_ransac = b
      inlier_mask = mask


  return k_ransac, b_ransac, inlier_mask
```

The inliers are computed with the following function:

Listing 2: RANSAC

```
def num_inlier(x,y,k,b,n_samples,thres_dist):
  num = 0
  mask = np.zeros(x.shape, dtype=bool)

  # distance point line
  line = k*x + b
```

```
    dist = np.abs(line-y) / np.sqrt(k**2 + 1)

    mask = dist < thres_dist

    num = len(dist[mask])

    return num, mask
```

A point is considered to be inlier if the its distance from the line is less then a threshold.

2.2 Results

In this section we are going to show the result for the RANSAC algorithm. The value for the k and be for the 3 methods are the following:

- $k_{gt} = 1,\ b_{gt} = 10$
- $k_{ls} = 0.615965657875546,\ b_{ls} = 8.961727141443642$
- $k_{RANSAC} = 0.9643894946568982,\ b_{RANSAC} = 9.982921294966832$

As shown above, the coefficient for the RANSAC are a very good estimation of the ground truth line, while the least squares is far away from the real line, due to the outliers present in the dataset.

These are the resulting lines that you can see in the figure below:

$$y_{gt} = x + 10$$
$$y_{ls} = 0.615965657875546 * x + 8.961727141443642$$
$$y_{RANSAC} = 0.9643894946568982 * x + 9.982921294966832$$

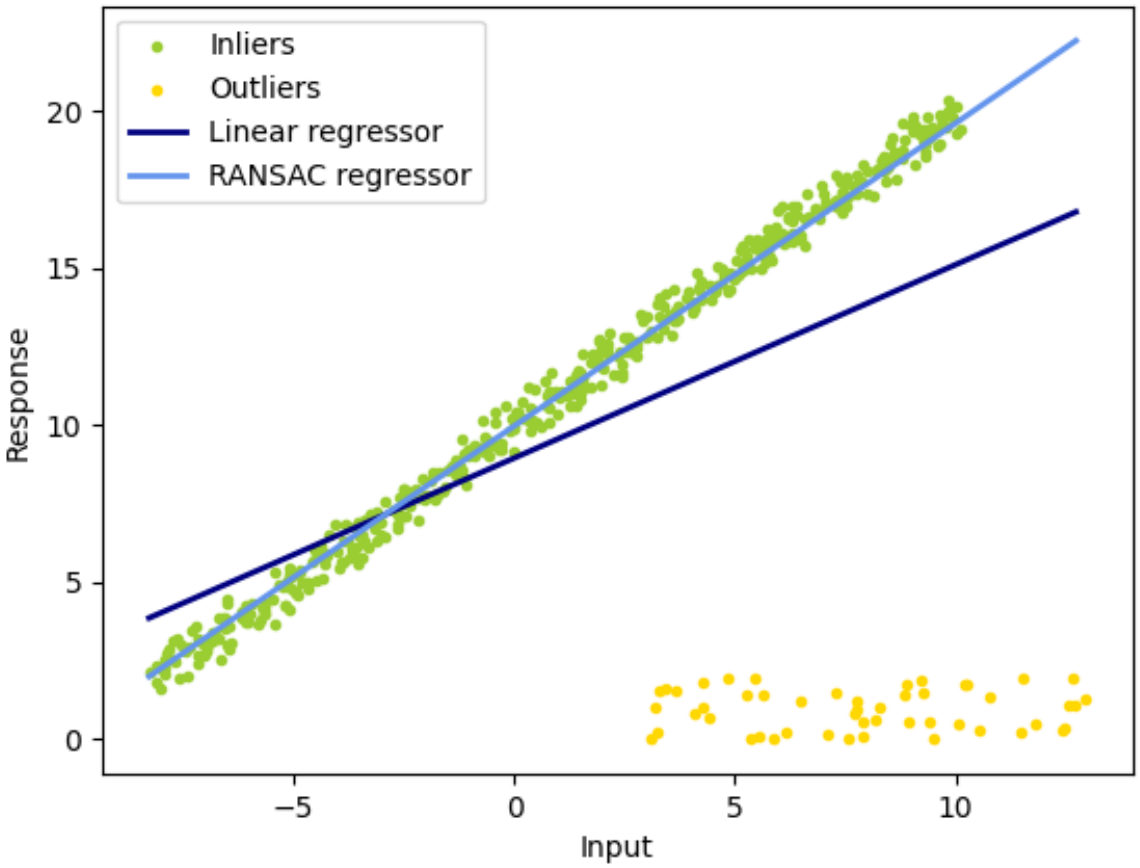

Figure 4: Model fitting results.