HW4 Bundle Adjustment

Yucong Chen 2019533079

Files:

> main.py
>
> vis.py
>
> bundle_adj.py
>
> add_noise.py

Usages:

> firstly run:
>
> python3 main.py
>
> then:
>
> python3 add_noise.py
>
> finally:
>
> python3 bundle_adj.py
>
> Visualization on point cloud can be found in vis.py

Task 1: Projection and Back-projection

projection:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & u_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

back-projection:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & u_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

Results:

World points:

x	-1.08	-0.54	0.00	0.54	1.08	-1.08	-0.54	0.00	0.54	1.08	-1.08	-0.54	0.00	0.54	1.08	-1.08	-0.54	0.00	0.54	1.08	-1.08	-0.54	0.00	0.54	1.08
y	-0.80	-0.80	-0.80	-0.80	-0.80	-0.40	-0.40	-0.40	-0.40	-0.40	0.00	0.00	0.00	0.00	0.00	0.40	0.40	0.40	0.40	0.40	0.80	0.80	0.80	0.80	0.80
z	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00

C1 view -- C9 view (C5 omitted):

u	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615
v	65	65	65	65	65	165	165	165	165	165	265	265	265	265	265	365	365	365	365	365	465	465	465	465	465

u	50	185	320	455	590	50	185	320	455	590	50	185	320	455	590	50	185	320	455	590	50	185	320	455	590
v	65	65	65	65	65	165	165	165	165	165	265	265	265	265	265	365	365	365	365	365	465	465	465	465	465

u	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565
v	65	65	65	65	65	165	165	165	165	165	265	265	265	265	265	365	365	365	365	365	465	465	465	465	465

u	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615
v	40	40	40	40	40	140	140	140	140	140	240	240	240	240	240	340	340	340	340	340	440	440	440	440	440

u	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565
v	40	40	40	40	40	140	140	140	140	140	240	240	240	240	240	340	340	340	340	340	440	440	440	440	440

u	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615	75	210	345	480	615
v	15	15	15	15	15	115	115	115	115	115	215	215	215	215	215	315	315	315	315	315	415	415	415	415	415

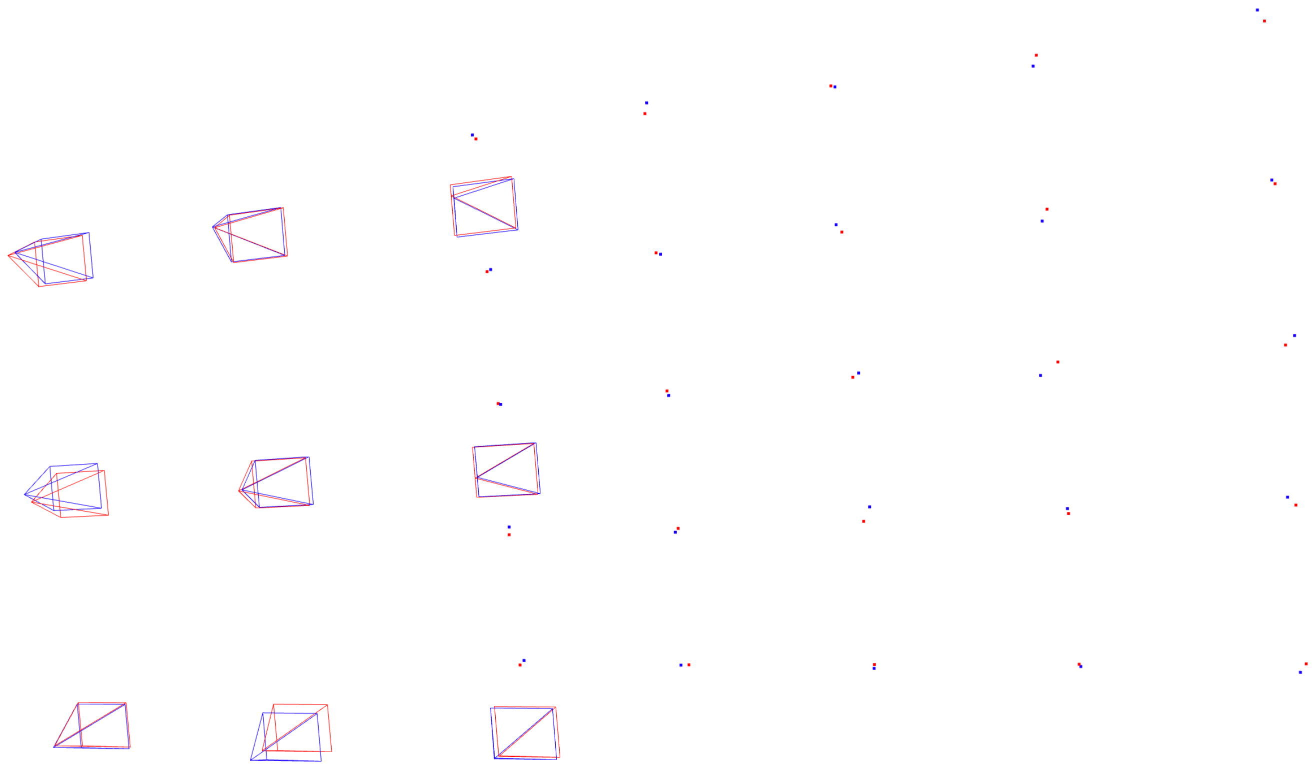
u	50	185	320	455	590	50	185	320	455	590	50	185	320	455	590	50	185	320	455	590	50	185	320	455	590
v	15	15	15	15	15	115	115	115	115	115	215	215	215	215	215	315	315	315	315	315	415	415	415	415	415

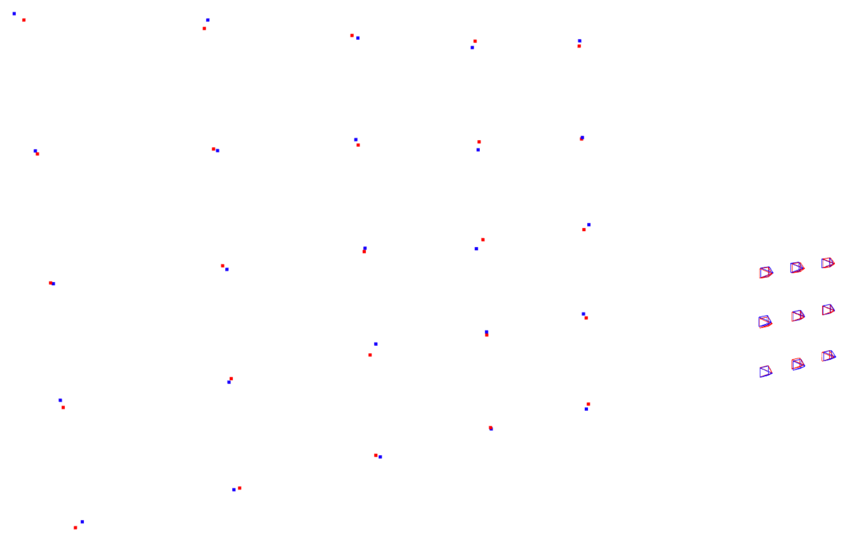
u	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565	25	160	295	430	565
v	15	15	15	15	15	115	115	115	115	115	215	215	215	215	215	315	315	315	315	315	415	415	415	415	415

Task 2: Add Noise

Add noise to world points and poses as follows (red is groundtruth and blue is perturbed data):

x	-1.06	-0.54	-0.00	0.56	1.06	-1.06	-0.54	-0.02	0.55	1.08	-1.10	-0.49	-0.01	0.53	1.07	-1.06	-0.52	0.01	0.53	1.08	-1.05	-0.53	-0.01	0.53	1.10
y	-0.82	-0.81	-0.80	-0.80	-0.78	-0.38	-0.39	-0.36	-0.41	-0.38	0.02	-0.03	0.01	-0.01	-0.00	0.41	0.37	0.42	0.39	0.41	0.82	0.78	0.79	0.83	0.82
z	2.00	2.00	1.97	1.99	1.99	2.00	2.00	1.99	2.00	2.01	1.98	1.98	2.02	1.99	2.00	1.98	1.99	1.99	2.00	2.02	1.96	2.01	1.98	1.99	2.02





And also add noise to the pixel measurement (which does not be visualized here)

Task 3: Bundle Adjustment

Calculate Jacobians Numerically:

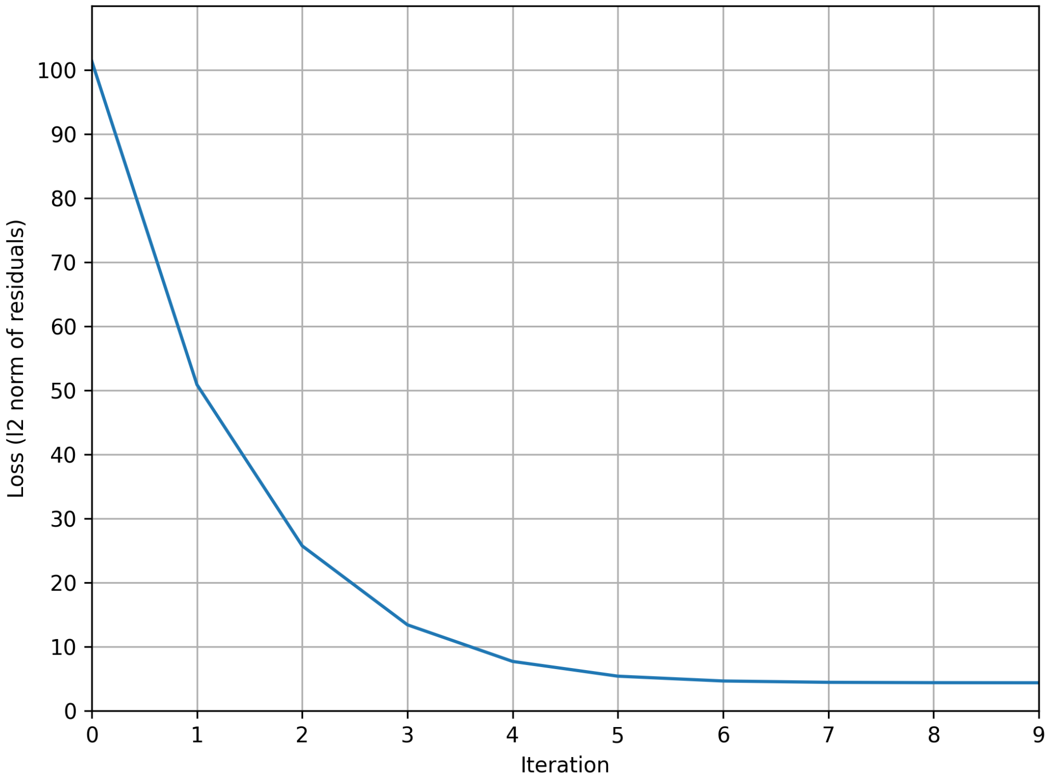
```
def Jacobian(input, f0):
    eps = 1e-10
    J = np.zeros((f0.shape[0], input.shape[0]))
    for i in range(input.shape[0]):
        increased_input = input.copy()
        increased_input[i] += eps
        f1 = f(increased_input)
        J[:, i] = (f1 - f0) / eps
    return J
```

Apply Gaussian-Newton Optimization:

```
def GN(X, input):
    for i in tqdm(range(10)):
        f0 = f(input)
        e0 = X - f0
        J = Jacobian(input, f0)
        H = J.T @ J
        delta = np.linalg.inv(H + np.eye(H.shape[0])) @ J.T @ e0
        input += 0.5*delta
    return input
```

Notice that I added identity matrix to H in the updating calculation to increase the stability and convergency. And also nomalized coordinates to eliminate K.

Results:



The curve shows the $||e_0||_2$, and its convergency after just few iterations.

The follows table shows the world points after doing bundle adjustment:

x	-1.08	-0.54	0.00	0.54	1.07	-1.08	-0.54	0.00	0.54	1.08	-1.07	-0.54	0.00	0.54	1.08	-1.08	-0.54	0.00	0.54	1.07	-1.07	-0.54	0.00	0.54	1.07
y	-0.80	-0.80	-0.79	-0.79	-0.79	-0.40	-0.40	-0.40	-0.39	-0.39	0.00	0.00	0.00	0.00	0.00	0.40	0.40	0.40	0.40	0.40	0.80	0.80	0.80	0.80	0.80
z	2.00	2.00	1.99	2.00	1.99	2.00	1.99	2.00	1.99	1.99	1.98	1.99	1.98	1.99	2.00	1.99	2.00	2.00	2.00	1.99	1.99	1.99	2.00	2.00	1.98

which is clearly more close to the groundtruth comparing with noisy data. And Here is the visualization:

where red is gt, blue is noisy data and green is bundle adjustment result.

Discussion:

- The original Gaussian-Newton method is unstable, and can be improved by adding an identity matrix to the J^TJ;
- The epsilon for numerical method is somehow important to maintain the stability;
- The learning rate is very important for both getting a good result and converging fast.