Optimisation-based IMU and Camera Integration

Youbing Wang

May 16, 2016

Abstract

Your abstract or summary can go here.

Contents

1 Introduction

In this work, we firstly simulate a navigation system equipped with an IMU and a RGB-D camera. Using the IMU's coordinates at the 1st pose as the global reference frame, the relative position of these two sensors at that time can be related by $\mathbf{A}_{u2c}$ and $\mathbf{T}_{u2c}$, :

$$\begin{aligned}\mathbf{A}_{u2c} &= (\alpha_{u2c}, \beta_{u2c}, \gamma_{u2c}),\\ \mathbf{T}_{u2c} &= (x_{u2c}, y_{u2c}, z_{u2c})\end{aligned}$$

And at the following poses, given IMU's states ($\mathbf{R}_{ui}$ and $\mathbf{T}_{ui}$), the camera's states ($\mathbf{R}_{ci}$ and $\mathbf{T}_{ci}$) can be obtained according to this formula:

$$\begin{aligned}\mathbf{R}_{ci} &= \mathbf{R}_{u2c}\mathbf{R}_{ui}\\ \mathbf{T}_{ci} &= \mathbf{T}_{u} + \mathbf{R}'_{ui}\mathbf{T}_{u2c}\end{aligned}$$

1.1 The Formulation of the Original Optimization Problem

For a system composed of an IMU and a RGB-D camera navigating with N camera poses and M features, the state vector $\mathbf{x}$ is defined as:

$$\mathbf{x} = (\overbrace{\mathbf{A}_{u2}, \mathbf{T}_{u2}, ..., \mathbf{A}_{uN}, \mathbf{T}_{uN}}^{(N-1)\times 6}, \overbrace{\mathbf{P}_{f1}, ..., \mathbf{P}_{fM}}^{M\times 3}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

where $\mathbf{A}_{ui} = (\alpha_{ui}, \beta_{ui}, \gamma_{ui})$, $\mathbf{T}_{ui} = (x_{ui}, y_{ui}, z_{ui})$, $\mathbf{P}_{fi} = (x_{fi}, y_{fi}, z_{fi})$, $d\mathbf{P}_i = (dx_i, dy_i, dz_i)$, $d\mathbf{v}_i = (dvx_i, dvy_i, dvz_i)$, and $d\mathbf{A}_i = (d\alpha_i, d\beta_i, d\gamma_i)$.

And the corresponding observations are:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_1, ..., \mathbf{uvd}_N, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9})'$$

where $\mathbf{uvd}_{ij} = (u_{ij}, v_{ij}, d_{ij})$ represents the image of the ith feature point at the jth camera pose.

1.2 Testing Methods

Since direct implementation could not obtain expected results and the initial results seem to indicate some kind of lack of restriction for the IMU data, we have decided to adopt the following method: firstly add some pseudo observations and at the same time reduce the IMU observations to enable the solution to be an over-constraint one, and then try to reduce those faked observations and add more IMU observations into the system step by step. Moreover, to ensure that our method (called *Jac* version thereafter) can always produce the right outcome, we have also implemented another version based on the standard implementation available in Matlab (called *Nonlin* version).

2 Testing Cases and Results

2.1 Test Version 1

2.1.1 Observation Vector:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, d\mathbf{p}_2, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.1.2 Comparison of Results by Two Methods:

Although both versions can converge to the ground truth values, *Jac* turns out to be much quicker. At the same time, it shows that the implementation of Jacobian part of $d\mathbf{p}_2$ with respect to $\mathbf{x}$ is correct.

2.2 Test Version 2

2.2.1 Observation Vector:

Add $d\mathbf{v}_2$ into $\mathbf{z}$,

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, d\mathbf{p}_2, d\mathbf{v}_2, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.2.2 Comparison of Results by Two Methods:

In most cases, both methods could converge. However, there are also scenarios that *Nonlin* could converge but *Jac* could not. I guess it is because that the former is a more complicated method with enhanced robustness. To verify this idea, I has tried to forbid *Nonlin* to use Gauss-Newton method by changing its options. And in this way the method appears to be more difficult to converge, which seems to agree with my point. An example is as follows,

Initial Value:

X0=[0.110256 0.356417 0.090190 -0.056813 -0.033857 -0.083328 0.063837 0.068673 0.067582 0.339370 -0.099879 -0.098265 0.142257 -0.308835 -0.034551 0.645063 -0.108795 0.079538 -1.511887 2.843616 0.026894 4.455303 2.931688 -0.118554 1.510241 5.692312 -3.066389 1.409409 5.848017 3.108783 0.077621 0.090106 0.162640 0.173855 0.046300 0.106586 0.186504 0.090145 -0.210427 0.385071 -0.164927 -0.304806 0.230750 -0.010454 -9.725180 -1.631339 -0.170097 -0.512165 1.604380 -0.047781 -0.032890 0.067601 -0.236558 -0.121574 0.191221 0.099644 -0.213547]

options =

lsqnonlin options:

Options used by current Algorithm ('levenberg-marquardt'): (Other available algorithms: 'trust-region-reflective')

Set by user: Algorithm: 'levenberg-marquardt' InitDamping: 0

Default: DerivativeCheck: 'off' Diagnostics: 'off' DiffMaxChange: Inf DiffMinChange: 0 Display: 'final' FinDiffRelStep: 'sqrt(eps)' FinDiffType: 'forward' FunValCheck: 'off' Jacobian: 'off' MaxFunEvals: '200*numberOfVariables' MaxIter: 400 OutputFcn: [] PlotFcns: [] ScaleProblem: 'none' TolFun: 1.0000e-06 TolX: 1.0000e-06 TypicalX: 'ones(numberOfVariables,1)'

Show options not used by current Algorithm ('levenberg-marquardt')

Solver stopped prematurely.

lsqnonlin stopped because it exceeded the function evaluation limit, options.MaxFunEvals = 11400 (the default value).

Final Value:

Xf=[0.110256 0.356417 0.090190 -0.056813 -0.033857 -0.083328 0.063837 0.068673 0.067582 0.339370 -0.099879 -0.098265 0.142257 -0.308835 -0.034551 0.645063 -0.108795 0.079538 -1.511887 2.843616 0.026894 4.455303 2.931688 -0.118554 1.510241 5.692312 -3.066389 1.409409 5.848017 3.108783 0.077621 0.090106 0.162640 0.173855 0.046300 0.106586 0.186504 0.090145 -0.210427 0.385071 -0.164927 -0.304806 0.230750 -0.010454 -9.725180 -1.631339 -0.170097 -0.512165 1.604380 -0.047781 -0.032890 0.067601 -0.236558 -0.121574 0.191221 0.099644 -0.213547]

maxe =

0.5122

2.3 Test Version 3

2.3.1 Observation Vector:

Add $d\mathbf{A}_2$ into $\mathbf{z}$,

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.3.2 Comparison of Results by Two Methods:

Similar to version 2.

2.4 Test Version 4

2.4.1 Observation Vector:

Add $d\mathbf{p}_i, d\mathbf{v}_i$ and $d\mathbf{A}_i$ at the following poses into $\mathbf{z}$,

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.4.2 Comparison of Results by Two Methods:

Given narrower range of noises, *Jac* could still converge after more iterations while *Nonlin* is still slow but more robust.

2.5 Test Version 5

2.5.1 Observation Vector:

Incorporate covariance matrix into *Nonlin*, then compare its results with those of *Jac*. Here $\mathbf{z}$ is the same as in Version 4:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.5.2 Comparison of Results by Two Methods:

Similar to version 2 and 3.

2.6 Test Version 6

2.6.1 Observation Vector:

In this test, the modification of the predicted is removed. Here $\mathbf{z}$ is the same as in Version 4 and 6:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.6.2 Effects:

Jac can converge much more quickly.

2.7 Test Version 7: Add Convariance Matrix into *Nonlin*

2.7.1 Observation Vector:

In this test, $\mathbf{z}$ is added with small noises, and its definition is the same as in Version 1:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, d\mathbf{p}_2, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

2.7.2 Comparison of Results by Two Methods:

The differences between them are within 1e-6. One example:

Jac: Ground Truth Value: Xg=[0.000000 -0.000000 0.000000 0.200000 0.000000 -0.000000 0.000000 -0.000000 0.000000 0.400000 0.000000 0.000000 0.000000 -0.000000 0.000000 0.600000 0.000000 -0.000000 -1.500000 3.000000 -0.000000 4.500000 3.000000 0.000000 1.500000 6.000000 -3.000000 1.500000 6.000000 3.000000 0.200000 0.000000 0.000000 0.200000 0.000000 0.000000 0.200000 0.000000 0.000000 0.200000 0.000000 0.000000 0.000000 0.000000 -9.800000 -1.570796

-0.000000 0.000000 1.500000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000] Initial Value: X0=[0.143987 -0.236152 0.033325 0.232638 -0.025189 0.148797 -0.060698 -0.137698 0.046002 0.476633 -0.036251 0.132687 -0.253004 -0.179153 -0.512028 0.686901 -0.165171 -0.042188 -1.331789 3.008502 -0.073405 4.358588 2.959938 0.100488 1.247291 5.988609 -2.869601 1.263212 6.111561 3.113873 0.052977 -0.250751 0.072190 0.334709 0.096490 0.127142 0.002979 0.097308 -0.096668 0.106439 -0.259316 0.183368 0.029176 0.167269 -9.548165 -1.760336 0.107170 -0.002127 1.652377 0.184616 0.136745 -0.136269 -0.021081 0.044024 0.526421 0.204436 0.386773]

ans =

-6.1304e-20

times=0 maxE = 1513.556603 maxDx = 4.278575 times=1 maxE = 839.256248 maxDx = 2.676602 times=2 maxE = 71.331826 maxDx = 1.994389 times=3 maxE = 80.173669 maxDx = 1.439257 times=4 maxE = 27.171264 maxDx = 0.432415 times=5 maxE = 4.898833 maxDx = 0.063140 times=6 maxE = 4.899495 maxDx = 0.000312 times=7 maxE = 4.899456 maxDx = 0.000002 times=8 maxE = 4.899456 maxDx = 0.000000 times=9 maxE = 4.899456 maxDx = 0.000000

Iteration Times: N = 9 Final Value: Xf=[-0.000003 -0.000000 -0.000022 0.199864 0.000231 -0.000207 -0.000001 0.000002 -0.000051 0.399683 0.000439 -0.000423 -0.000003 0.000001 -0.000266 0.598412 0.000700 -0.000630 -1.504583 2.998154 0.003533 4.495345 3.002784 -0.002862 1.489867 6.001621 -2.998428 1.496255 5.999394 3.001582 0.199485 -0.000477 -0.000421 0.199552 -0.001551 0.000930 0.200902 0.000138 -0.000378 0.200143 0.001605 0.001349 -0.000453 0.000168 -9.800305 -1.570395 0.001065 0.000900 1.498469 0.000505 -0.000864 -0.000372 0.000788 -0.000516 -0.000160 0.000591 0.001634]

maxe =

0.0101

Nonlin:

Initial Value: X0=[0.143987 -0.236152 0.033325 0.232638 -0.025189 0.148797 -0.060698 -0.137698 0.046002 0.476633 -0.036251 0.132687 -0.253004 -0.179153 -0.512028 0.686901 -0.165171 -0.042188 -1.331789 3.008502 -0.073405 4.358588 2.959938 0.100488 1.247291 5.988609 -2.869601 1.263212 6.111561 3.113873 0.052977 -0.250751 0.072190 0.334709 0.096490 0.127142 0.002979 0.097308 -0.096668 0.106439 -0.259316 0.183368 0.029176 0.167269 -9.548165 -1.760336 0.107170 -0.002127 1.652377 0.184616 0.136745 -0.136269 -0.021081 0.044024 0.526421 0.204436 0.386773]

Local minimum found.

Optimization completed because the size of the gradient is less than the default value of the function tolerance.

¡stopping criteria details¿

Final Value: Xf=[-0.000003 -0.000000 -0.000022 0.199864 0.000231 -0.000207 -0.000001 0.000002 -0.000051 0.399683 0.000439 -0.000423 -0.000003 0.000001 -0.000266 0.598412 0.000700 -0.000630 -1.504583 2.998154 0.003533 4.495345 3.002784 -0.002862 1.489867 6.001621 -2.998428 1.496255 5.999394 3.001582 0.199485 -0.000477 -0.000421 0.199552 -0.001551 0.000930 0.200902 0.000138 -0.000378 0.200143 0.001605 0.001349 -0.000453 0.000168 -9.800305 -1.570395 0.001065 0.000900 1.498469 0.000505 -0.000864 -0.000372 0.000788 -0.000516 -0.000160 0.000591 0.001634]

maxe =

0.0101

2.7.3 Test Based on Version 6:

For test version 6, if I manually change all negative elements of the covariance matrix into zero, I can also obtain similar results.

2.8 Test Version 8

2.8.1 Observation Vector:

In this test, the pseudo observations of bf and bw removed. Here $\mathbf{z}$ is the same as in Version 4 and 6:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c})'$$

2.8.2 Effects:

Jac can converge much more quickly.

2.9 Test Version 9

2.9.1 Observation Vector:

Based on version 8, all pseudo observations of $\mathbf{v}_i$s are removed, therefore,

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c})'$$

2.9.2 Effects:

Jac can converge, but its final results become further to the ground truth values compared with the previous step.

2.10 Test Version 10

2.10.1 Observation Vector:

The observation vector is the same as that of version 9:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c})'$$

In this section, we are trying to check its performance while taking different routes and processes.

2.10.2 Constant Velocity Along a Line:

Simulated Scenario: the system moves along a line with a constant speed of 0.2 m/s, the key frames of the camera are taken at 1 s intervals, and the IMU has a sampling rate of 600 Hz. The relative position of the camera in IMU's coordinates is: $\mathbf{A}_{u2c} = (0, 0, -\pi/2)', \mathbf{T}_{u2c} = (1.5, 0, 0)'$.

Results: *Jac* can converge easily.

2.10.3 Changing Velocity Along Sections of Lines in Different Planes:

Simulated Scenario: the system moves along several lines. The key frames of the camera are taken at 1 s intervals. At each key frame of, its speed is zero. Between every two key frames, the system experience an acceleration process followed by a damping one. And the IMU has a sampling rate of 600 Hz. The relative position of the camera in IMU's coordinates is: $\mathbf{A}_{u2c} = (0, 0, -\pi/2)', \mathbf{T}_{u2c} = (1.5, 0, 0)'$.

Results: *Jac* can converge easily.

2.11 Test Version 11

2.11.1 Observation Vector:

g is removed from the observation vector:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c})'$$

2.11.2 The Special Routes that Make the Jacobian Matrix Non-singular:

During the simulation, I find that I need to make the system rotation around the pitch and roll axises to make the Jacobian matrix non-singular.

2.12 Test Version 12

2.12.1 Observation Vector:

$\mathbf{A}_{u2c}$ and $\mathbf{T}_{u2c}$ are removed from the observation vector:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9})'$$

2.12.2 The Special Routes:

During the simulation, I find that along a helix route, if the system keeps rotating around the pitch and roll axises, the corresponding Jacobian matrix will not be singular.

2.13 Test Version 13: Minimum Number of Features Needed

2.13.1 Observation Vector:

The observation vector is the same as that in version 12:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9})'$$

2.13.2 The Special Scenario:

I find that along a circle route, if the system keeps rotating around the pitch and roll axises, the corresponding Jacobian matrix will not be singular when the pose number is 9 and only 1 feature is situated in the scenario.

2.14 Test Version 14: Improve the Pre-integration Algorithm

2.14.1 Observation Vector:

The observation vector is the same as that in version 12 and 13:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9})'$$

2.14.2 The Simulated Scenario:

The simulated route is the same as that of version 13: while moving along a circle route, the system keeps rotating around the pitch and roll axes, and the pose number is 9 and only 1 feature is situated in the scenario.

2.14.3 The Original Pre-integration Method

The Original Pre-integration: in Todd's TRO paper [Lupton and Sukkarieh(2012)], the pre-integration is carried out according to the following formulas:

$$\begin{aligned}
\triangle t &= t_{t+1} - t_t \\
f_t^{bt1} &= R_{bt}^{bt1}(f_t^b - b_f) \\
\triangle v_{t+1} &= \triangle v_t + f_t^{bt1}\triangle t \\
\triangle p_{t+1}^{+} &= \triangle p_t^{+} + \triangle v_t \triangle t \\
\triangle \mathbf{A}_{t+1} &= \triangle \mathbf{A}_t + E_{bt}^{bt1}(\omega_t^b - b_\omega)\triangle t
\end{aligned}$$

The Corresponding Calculation of Covariance:

The covariance is computed in the following way:

$$
\begin{aligned}
\frac{\partial \triangle p_{t+1}^{+}}{\partial \triangle p_t} &= \mathbf{I}_3 \\
\frac{\partial \triangle p_{t+1}^{+}}{\partial \triangle v_t} &= \mathbf{I}_3 \triangle t \\
\frac{\partial \triangle v_{t+1}}{\partial \triangle v_t} &= \mathbf{I}_3 \\
\frac{\partial \triangle v_{t+1}}{\partial \triangle \mathbf{A}_t} &= \alpha \triangle t \\
\frac{\partial \triangle v_{t+1}}{\partial \triangle b_{f(t)}} &= -R_{bt}^{bt1} \triangle t \\
\frac{\partial \triangle \mathbf{A}_{t+1}}{\partial \triangle \mathbf{A}_t} &= \mathbf{I}_3 + \beta \triangle t \\
\frac{\partial \triangle \mathbf{A}_{t+1}}{\partial \triangle b_{\omega(t)}} &= -E_{bt}^{bt1} \triangle t \\
\frac{\partial \triangle b_{f(t+1)}}{\partial \triangle b_{f(t)}} &= \mathbf{I}_3 \\
\frac{\partial \triangle b_{\omega(t+1)}}{\partial \triangle b_{\omega(t)}} &= \mathbf{I}_3
\end{aligned}
$$

2.14.4 Proposed Modification to the Pre-integration Method

Pre-integration: to improve the accuracy of the pre-integration results, we proposed to change the calculation of $\triangle p_t^{+}$ according to the following formula:

$$
\begin{aligned}
\triangle p_{t+1}^{+} &= \triangle p_t^{+} + (\triangle v_{t+1} + \triangle v_t)\triangle t/2 \\
&= \triangle p_t^{+} + \triangle v_t \triangle t + f_t^{bt1} (\triangle t)^2/2
\end{aligned}
$$

Accordingly, $\triangle p_{t+1}^{+}$ is also connected with $\mathbf{A}_t$ and b_f as well:

$$
\begin{aligned}
\frac{\partial \triangle p_{t+1}^{+}}{\partial \triangle \mathbf{A}_t} &= \alpha (\triangle t)^2/2 \\
\frac{\partial \triangle p_{t+1}^{+}}{\partial \triangle b_{f(t)}} &= -R_{bt}^{bt1} (\triangle t)^2/2
\end{aligned}
$$

2.14.5 Further Possible Modification to the Pre-integration Method

To further improve the accuracy of the pre-integration results, we proposed to change the calculation of $\triangle p_{t+1}^{+}$ according to the following formula:

$$\begin{aligned}
\triangle v_{t+1} &= \triangle v_t + (f(\triangle \mathbf{A}_t)_t^{bt1} + f(\triangle \mathbf{A}_{t+1})_t^{bt1}) * \triangle t \\
\triangle p_{t+1}^{+} &= \triangle p_t^{+} + (\triangle v_{t+1} + \triangle v_t)\triangle t/2 \\
&= \triangle p_t^{+} + \triangle v_t * \triangle t + f_t^{bt1}(\triangle t)^2/2
\end{aligned}$$

Accordingly, the relation between $\triangle v_{t+1}^{+}$ and $\mathbf{A}_t$, b_f has changed:

2.15 Average Data Based Modification to the Pre-integration Method

In this method, the IMU data is averaged between every two consecutive time steps. And then the calculation of $\triangle p_{t+1}^{+}, \triangle v_{t+1}$, $\triangle \mathbf{A}_{t+1}$ and their corresponding covariance matrix is still performed according to the original formulas. More specifically, the implemented Matlab code is as follows:

$$dataIMU(1:(end-1), 2:7) = 0.5(dataIMU(1:(end-1), 2:7) + dataIMU(2:end, 2:7))$$

On the other hand, Q, which reflects the noise level of IMU data, has been changed according to the corresponding magnitude of pseudo noise added: $Q = eye(15) \times fnoiselevel \times fnoiselevel$ However, the results remain the same.

2.16 Evaluating the Pre-integration Method

Based on the results of the previous simulations, we feel that pre-integration as a relatively new method has its limitations which have not been fully explored. Coupled with the observability problem of the related parameters, what kind of methods and visual-inertial navigation system (VINS) data can practically enable us to obtain a unique and as accurate as possible solution is still an unknown issue. And this constitutes the sole purpose of our work at this section. By constructing a formulation of the original VINS problem and then exploring possible solution to it, we can establish a practical touch stone to test the performance of all other approximation based methods including the pre-integration one.

In this section, the IMU data at each sampling time step are used as a new observation. For a synchronised system equipped with an IMU sampling at K Hz rate, the corresponding observation vector $\mathbf{z}$ is composed as follows:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3}, \overbrace{\omega_{02}, a_{02}, ..., \omega_{(K-1)2}, a_{(k-1)2}, ..., \omega_{(K-1)N}, a_{(K-1)N}}^{K\times(N-1)\times 6})'$$

However, taking the asynchronous nature of IMU and camera readings into account, $\mathbf{z}$ is composed as follows:

$$\mathbf{z} = (\overbrace{\mathbf{uvd}_{11}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N_{poses}}, ..., \mathbf{uvd}_{MN_{poses}}}^{M\times N_{poses}\times 3}, \overbrace{\omega_1, a_1, \mathbf{0}_1 ..., \omega_{F(N_{poses}-1)}, a_{F(N_{poses}-1)}, \mathbf{0}_{F(N_{poses}-1)}}^{F\times(N_{poses}-1)\times 9})'$$

where T represents the total time span in seconds between the N camera poses. In this case, the camera poses need to be obtained through interpolation.

Accordingly, the state vector $\mathbf{x}$ becomes:

$$\mathbf{x} = (\overbrace{\mathbf{A}_{u2}, \mathbf{T}_{u2}, ..., \mathbf{A}_{u(TF+1)}, \mathbf{T}_{u(TF+1)}}^{T\times F\times 6}, \overbrace{\mathbf{P}_{f1}, ..., \mathbf{P}_{fM}}^{M\times 3}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_{TF+1}}^{(T\times F+1)\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

Given the number of camera key frames N_{poses}, then $T = N_{poses} - 1$ and the state vector $\mathbf{x}$ can be written as:

$$\mathbf{x} = (\overbrace{\mathbf{A}_{u2}, \mathbf{T}_{u2}, ..., \mathbf{A}_{u((N_{poses}-1)F+1)}, \mathbf{T}_{u((N_{poses}-1)F+1)}}^{(N_{poses}-1)\times F\times 6}, \overbrace{\mathbf{P}_{f1}, ..., \mathbf{P}_{fM}}^{M\times 3}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_{(N_{poses}-1)F+1}}^{((N_{poses}-1)\times F+1)\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)'$$

$$\begin{aligned}
\dot{\mathbf{v}}_i &= C_i^0(a_i - b_f) + g^0 \\
\dot{\mathbf{A}}_i &= E_i^0(\omega_i - b_w) \\
\dot{T}_i &= \mathbf{v}_i
\end{aligned}$$

And their corresponding discrete forms are:

$$\begin{aligned}
\frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{\triangle t} &= C_i^0(a_i - b_f) + g^0 \\
\frac{\mathbf{A}_{i+1} - \mathbf{A}_i}{\triangle t} &= E_i^0(\omega_i - b_w) \\
\frac{\mathbf{T}_{i+1} - \mathbf{T}_i}{\triangle t} &= \mathbf{v}_i
\end{aligned}$$

Therefore, the IMU data related prediction functions are:

$$\begin{aligned}
a_i &= C_0^i(\frac{\mathbf{v}_{i+1} - \mathbf{v}_i}{\triangle t} - g^0) + b_f \\
\omega_i &= E_0^i(\frac{\mathbf{A}_{i+1} - \mathbf{A}_i}{\triangle t}) + b_w \\
0 &= \mathbf{T}_{i+1} - \mathbf{T}_i - \mathbf{v}_i \triangle t
\end{aligned}$$

On the other hand, Q, which reflects the noise level of IMU data, has been changed according to the corresponding magnitude of pseudo noise added: $Q = eye(15) \times fnoiselevel \times fnoiselevel$ However, the results become ever worse.

3 Formulation of The Problem

In this section, we will formulate the simulated problem in a comprehensive way, so that the readers could fully understand our code.

3.1 The Raw Data From the Sensors

In our simulation, a camera and an IMU are employed. As the result, the raw data is made of the raw camera data and IMU data.

3.1.1 The Raw Camera Data

The raw camera data is composed of images of features, i.e., $uv = (u, v)$ for a mono-camera, or $uvd = (u, v, d)$ if the camera is a RGB-D camera. If there are M features that could be seen at N camera poses, all of the observed raw camera data can be written as:

$$\mathbf{z}_{camera} = (\overbrace{\mathbf{uv}_{11}, \mathbf{uv}_{21}, ..., \mathbf{uv}_{M1}, ..., \mathbf{uv}_{1N}, \mathbf{uv}_{2N}, ..., \mathbf{uv}_{MN}}^{M\times N\times 2})' \tag{1}$$

where $\mathbf{uv}_{ij} = (u_{ij}, v_{ij})$ represents the image of the ith feature point taken at the jth mono-camera pose.

Or, for a RGB-D camera,

$$\mathbf{z}_{camera} = (\overbrace{\mathbf{uvd}_{11}, \mathbf{uvd}_{21}, ..., \mathbf{uvd}_{M1}, ..., \mathbf{uvd}_{1N}, \mathbf{uvd}_{2N}, ..., \mathbf{uvd}_{MN}}^{M\times N\times 3})' \tag{2}$$

where $\mathbf{uvd}_{ij} = (u_{ij}, v_{ij}, d_{ij})$ represents the image of the ith feature point taken at the jth camera pose.

In this simulation, to simplify the pre-processing steps, we assume that all features can be observed at every key camera pose.

3.1.2 The Raw IMU Data

IMUs sample the angular velocity and acceleration at a higher data rate (ranging from 70 to 1k Hz). Therefore, their data are made of $(\omega_i, \mathbf{a}_i)$ (where $\omega_i = (\omega_{xi}, \omega_{yi}, \omega_{zi})'$, $\mathbf{a}_i = (a_{xi}, a_{yi}, a_{zi})'$) from each time step. For an IMU with a data sampling rate of K (represented as $nIMUrate$ in the code) running between N camera key frames (for simplicity, we assume those sequential key frames are grabbed once per second), we can write the corresponding raw IMU data as:

$$\mathbf{z}_{IMUraw} = (\overbrace{\omega\mathbf{a}_{01}, \omega\mathbf{a}_{11}, ..., \omega\mathbf{a}_{(K-1)1}, ..., \omega\mathbf{a}_{0(N-1)}, \omega\mathbf{a}_{1(N-1)}, ..., \omega\mathbf{a}_{(K-1)(N-1)}}^{K\times(N-1)\times 6})' \tag{3}$$

where $\omega\mathbf{a}_{ij} = (\omega_{ij}, \mathbf{a}_{ij})$ represents the ith sampled IMU data (starting from 0 and ending at $(K-1)$) between the jth and $(j+1)$th camera pose (corresponding to the jth and $(j+1)$th second in our case).

3.2 The Noises Added to the Data Computed From Ground Truth

We have simulated a route for the system ensuring that there are enough motions along different axes. Based on the scenario, we can obtain the ground truth values of the camera and IMU data. Then we can add Gaussian noises to them to generate noisy raw data. For the camera, we assume that for u, v, their noise level is $\sigma_{uv} = 1pixel$. And for d, $\sigma_d = 0.1m$. while for the IMU, we simply adopt the value provided in the TRO paper [Lupton and Sukkarieh(2012)], i.e., $\sigma_\omega = 0.001rad/s$ and $\sigma_{\mathbf{a}} = 0.0775m/s^2$.

3.3 The Non-pre-integration Method

In the non-pre-integration method, both the raw camera data and raw IMU data are regarded as observations.

3.3.1 The State Vector x

The state vector is made up of the IMU rotation (the corresponding Euler angles are represented as $\mathbf{A}_i$)/translation ($\mathbf{T}_i$)/velocity ($\mathbf{v}_i$) ($i = 1, ..., K(N-1)+1$), feature positions ($\mathbf{P}_{fi}, (i = 1, ..., M)$), gravity $\mathbf{g}$ in the initial frame, the relative position between the camera and the IMU (the corresponding Euler angles are represented as $\mathbf{A}_{u2c}$, translation is $\mathbf{T}_{u2c}$), and the biases ($\mathbf{b}_f$ for acceleration and $\mathbf{b}_\omega$ for rotation velocity bias).

More specifically, for a system composed of an IMU with a data sampling rate of K Hz and a camera navigating with N key camera poses (corresponding to N seconds) and M features, the state vector $\mathbf{x}$ is defined as:

$$\mathbf{x} = (\overbrace{\mathbf{A}_{11}, \mathbf{T}_{11}, ..., \mathbf{A}_{0N}, \mathbf{T}_{0N}}^{K(N-1)\times 6}, \overbrace{\mathbf{P}_{f1}, ..., \mathbf{P}_{fM}}^{M\times 3}, \overbrace{\mathbf{v}_{01}, ..., \mathbf{v}_{0N}}^{(K(N-1)+1)\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w)' \tag{4}$$

where $\mathbf{A}_{ij} = (\alpha_{ij}, \beta_{ij}, \gamma_{ij})$, $\mathbf{T}_{ij} = (x_{ij}, y_{ij}, z_{ij})$, $\mathbf{P}_{fi} = (x_{fi}, y_{fi}, z_{fi})$, $\mathbf{g} = (g_x, g_y, g_z)$, $\mathbf{A}_{u2c} = (\alpha_{u2c}, \beta_{u2c}, \gamma_{u2c})$, $\mathbf{T}_{u2c} = (x_{u2c}, y_{u2c}, z_{u2c})$, $\mathbf{b}_f = (b_{fx}, b_{fy}, b_{fz})$, and $\mathbf{b}_\omega = (b_{\omega x}, b_{\omega y}, b_{\omega z})$.

And given IMU's states (R_i and $\mathbf{T}_i$), the camera's states (R_{ci} and $\mathbf{T}_{ci}$) can be obtained according to the following formulas:

$$\begin{aligned}
R_i &= R_x(\alpha_i)R_y(\beta_i)R_z(\gamma_i) \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_i) & \sin(\alpha_i) \\ 0 & -\sin(\alpha_i) & \cos(\alpha_i) \end{pmatrix} \begin{pmatrix} \cos(\beta_i) & 0 & -\sin(\beta_i) \\ 0 & 1 & 0 \\ \sin(\beta_i) & 0 & \cos(\beta_i) \end{pmatrix} \begin{pmatrix} \cos(\gamma_i) & \sin(\gamma_i) & 0 \\ -\sin(\gamma_i) & \cos(\gamma_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} && (5) \\
R_{u2c} &= R_x(\alpha_{u2c})R_y(\beta_{u2c})R_z(\gamma_{u2c}) && (6) \\
R_{ci} &= R_{u2c}R_i && (7) \\
\mathbf{T}_{ci} &= \mathbf{T}_i + R_i'\mathbf{T}_{u2c} && (8)
\end{aligned}$$

3.3.2 The Observation Vector

In the observation vector, in addition to the raw camera data and raw IMU data we have mentioned before, the relations between translation and velocity, i.e., $\mathbf{0} = \mathbf{T}_{i+1} - \mathbf{T}_i - \mathbf{v}_i \triangle t$, are also added. Consequently, the observation vector $\mathbf{z}$ is

$$\begin{aligned}
\mathbf{z}_{raw} &= (\mathbf{z}_{camera}, \mathbf{z}_{IMUraw}, \mathbf{z}_{Tv})' \\
&= (\overbrace{\mathbf{uv}_{11}, \mathbf{uv}_{21}, ..., \mathbf{uv}_{M1}, ..., \mathbf{uv}_{1N}, \mathbf{uv}_{2N}, ..., \mathbf{uv}_{MN}}^{M\times N\times 2}, \\
&\overbrace{\omega\mathbf{a}_{01}, \omega\mathbf{a}_{11}, ..., \omega\mathbf{a}_{(K-1)1}, ..., \omega\mathbf{a}_{0(N-1)}, \omega\mathbf{a}_{1(N-1)}, ..., \omega\mathbf{a}_{(K-1)(N-1)}}^{K\times(N-1)\times 6}, \quad (9) \\
&\overbrace{\mathbf{0}, \mathbf{0}, ..., \mathbf{0}}^{K\times(N-1)\times 3})' \quad (10)
\end{aligned}$$

3.3.3 The Observation Function $H(\mathbf{x})$

For the features observed through the camera, the following formulas hold:

$$\begin{aligned}
\mathbf{P}_{ij} &= (x_{ij}, y_{ij}, z_{ij}) = R_{cj}(\mathbf{P}_{fi} - \mathbf{T}_{c0j}) \quad (11) \\
u_{ij} &= f * x_{ij}/z_{ij} + cx_0 \quad (12) \\
v_{ij} &= f * y_{ij}/z_{ij} + cy_0 \quad (13) \\
d_{ij} &= z_{ij} \quad (14)
\end{aligned}$$

where f is the focal length of the camera, (cx_0, cy_0) is the displacement of the origin of the camera.

On the other hand, the part about the observations of IMU in $H(\mathbf{x})$ can be broken into the following three parts:

$$\begin{aligned}
\omega_{ij} &= E_{ij}(\mathbf{A}_{(i+1)j} - \mathbf{A}_{ij})/\triangle t + \mathbf{b}_w \quad (15) \\
\mathbf{a}_{ij} &= R_{ij}((\mathbf{v}_{(i+1)j} - \mathbf{v}_{ij})/\triangle t - \mathbf{g}) + \mathbf{b}_f \quad (16) \\
\mathbf{bZeros} &= \mathbf{T}_{(i+1)j} - \mathbf{T}_{ij} - \mathbf{v}_{ij}\triangle t \quad (17)
\end{aligned}$$

where $i = 0, ..., K-1$, and $R_{ij}, E_{ij}(= \begin{pmatrix} 1 & 0 & -\sin(\beta_{ij}) \\ 0 & \cos(\alpha_{ij}) & \cos(\beta_{ij})\sin(\alpha_{ij}) \\ 0 & -\sin(\alpha_{ij}) & \cos(\beta_{ij})\cos(\alpha_{ij}) \end{pmatrix})$ correspond to the rotation matrix and rotation rate matrix for the IMU at the time step i since the jth key camera frame respectively.

Putting all items together, $H(\mathbf{x})$ can be written as:

$$
\begin{aligned}
H(\mathbf{x}) = {} & (H_{camera}(\mathbf{x}), H_{IMUraw}(\mathbf{x}), H_{Tv}(\mathbf{x})) \\
= & (\overbrace{u_{11}, v_{11}, ..., u_{M1}, v_{M1}, ..., u_{1N}, v_{1N}, ..., u_{MN}, v_{MN}}^{M\times N\times 2}, \\
& \overbrace{\omega\mathbf{a}_{01}, \omega\mathbf{a}_{11}, ..., \omega\mathbf{a}_{(K-1)1}, ..., \omega\mathbf{a}_{0(N-1)}, \omega\mathbf{a}_{1(N-1)}, ..., \omega\mathbf{a}_{(K-1)(N-1)}}^{K\times(N-1)\times 6}, \\
& \overbrace{\mathbf{T}_2 - \mathbf{T}_1 - \mathbf{v}_1\triangle t, \mathbf{T}_3 - \mathbf{T}_2 - \mathbf{v}_2\triangle t, ..., \mathbf{T}_{(N-1)K+1} - \mathbf{T}_{(N-1)K} - \mathbf{v}_{(N-1)K}\triangle t}^{K\times(N-1)\times 3}) \\
= & (\overbrace{f * x_{11}/z_{11} + cx_0, f * y_{11}/z_{11} + cy_0, ..., f * x_{MN}/z_{MN} + cx_0, f * y_{MN}/z_{MN} + cy_0,}^{M\times N\times 2}, \\
& \overbrace{E_i * (\mathbf{A}_{11} - \mathbf{A}_{01})/\triangle t + \mathbf{b}_w, ..., R_{(K-1)(N-1)} * ((\mathbf{v}_{0N} - \mathbf{v}_{(K-1)(N-1)})/\triangle t - \mathbf{g}) + \mathbf{b}_f}^{K\times(N-1)\times 6}, \\
& \overbrace{\mathbf{T}_2 - \mathbf{T}_1 - \mathbf{v}_1\triangle t, \mathbf{T}_3 - \mathbf{T}_2 - \mathbf{v}_2\triangle t, ..., \mathbf{T}_{(N-1)K+1} - \mathbf{T}_{(N-1)K} - \mathbf{v}_{(N-1)K}\triangle t}^{K\times(N-1)\times 3})
\end{aligned}
\tag{18}
$$

3.3.4 Jacobian Matrix

Based on the composition of $H(\mathbf{x})$, the corresponding Jacobian matrix can be calculated.

For camera observations of (u_{ij}, v_{ij}, d_{ij}) which represent the observation of the ith feature at the jth camera pose,

$$\frac{\partial u_{ij}}{\partial \mathbf{P}_{ij}} = [f/z_{ij}, 0, -fx_{ij}/z_{ij}^2] \tag{19}$$

$$\frac{\partial v_{ij}}{\partial \mathbf{P}_{ij}} = [0, f/z_{ij}, -fy_{ij}/z_{ij}^2] \tag{20}$$

$$\frac{\partial d_{ij}}{\partial \mathbf{P}_{ij}} = [0, 0, 1] \tag{21}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{A}_{0j}} = R_{u2c}\frac{\partial R_{0j}}{\partial \mathbf{A}_{0j}}(\mathbf{P}_{i1} - \mathbf{T}_{0j}) \tag{22}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{T}_{0j}} = -R_{u2c}R_{0j} \tag{23}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{A}_{u2c}} = \frac{\partial R_{u2c}}{\partial \mathbf{A}_{u2c}}R_{0j}(\mathbf{P}_{i1} - R'_{0j}\mathbf{T}_{u2c} - \mathbf{T}_{0j}) \tag{24}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{T}_{u2c}} = -R_{u2c} \tag{25}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{P}_{fi}} = R_{u2c}R_{0j} \tag{26}$$

For ω_{ij},

$$\begin{aligned}
\frac{\partial \omega_{ij}}{\partial \mathbf{A}_{ij}} &= \frac{\partial E_{ij}}{\partial \mathbf{A}_{ij}}(\mathbf{A}_{(i+1)j} - \mathbf{A}_{ij})/\triangle t + E_{ij}(-\frac{\partial \mathbf{A}_{ij}}{\partial \mathbf{A}_{ij}})/\triangle t \\
&= (\frac{\partial E_{ij}}{\partial \mathbf{A}_{ij}}(\mathbf{A}_{(i+1)j} - \mathbf{A}_{ij}) - E_{ij})/\triangle t && (27) \\
\frac{\partial E_{ij}}{\partial \mathbf{A}_{ij}} &= [\frac{\partial E_{ij}}{\partial \alpha_{ij}}, \frac{\partial E_{ij}}{\partial \beta_{ij}}, \frac{\partial E_{ij}}{\partial \gamma_{ij}}] && (28) \\
\frac{\partial \omega_{ij}}{\partial \mathbf{A}_{(i+1)j}} &= E_{ij}\frac{\partial \mathbf{A}_{(i+1)j}}{\partial \mathbf{A}_{(i+1)j}}/\triangle t && (29) \\
&= E_{ij}/\triangle t && (30) \\
\frac{\partial \omega_{ij}}{\partial b_\omega} &= I_{3\times 3} && (31)
\end{aligned}$$

For $\mathbf{a}_{ij}$,

$$\begin{aligned}
\frac{\partial \mathbf{a}_{ij}}{\partial \mathbf{A}_{ij}} &= \frac{\partial R_{ij}}{\mathbf{A}_{ij}}((\mathbf{v}_{i+1} - \mathbf{v}_i)/\triangle t - \mathbf{g}) && (32) \\
\frac{\partial R_{ij}}{\partial \mathbf{A}_{ij}} &= [\frac{\partial R_{ij}}{\partial \alpha_{ij}}, \frac{\partial R_{ij}}{\partial \beta_{ij}}, \frac{\partial R_{ij}}{\partial \gamma_{ij}}] && (33) \\
\frac{\partial \mathbf{a}_{ij}}{\partial \mathbf{v}_{(i+1)j}} &= R_{ij}/\triangle t && (34) \\
\frac{\partial \mathbf{a}_{ij}}{\partial \mathbf{v}_{ij}} &= -R_{ij}/\triangle t && (35) \\
\frac{\partial \mathbf{a}_{ij}}{\partial \mathbf{g}} &= -R_{ij} && (36) \\
\frac{\partial \mathbf{a}_{ij}}{\partial b_f} &= I_{3\times 3} && (37)
\end{aligned}$$

For $\mathbf{bZeros}_{ij}$,

$$\begin{aligned}
\frac{\partial \mathbf{bZeros}_{ij}}{\partial \mathbf{T}_{(i+1)j}} &= I_{3\times 3} && (38) \\
\frac{\partial \mathbf{bZeros}_{ij}}{\partial \mathbf{T}_{ij}} &= -I_{3\times 3} && (39) \\
\frac{\partial \mathbf{bZeros}_{ij}}{\partial \mathbf{v}_{ij}} &= -I_{3\times 3}\triangle t && (40)
\end{aligned}$$

3.3.5 The Corresponding Code Functions

In the Matlab code, the raw camera data and IMU data are generated by calling a simulation function *fnSimIMUnFeaturesAtNPoses_helix*, and the noise are added through Line 139 to 162 in the *Main_simuNpose.m*. The state vector x is made up through Line 185 to 250 in this file. Moreover, the observation vector is represented as *Zobs* and is composed from Line

273 to 357 in the same file. On the other hand, *fnCnUPredErr_lsqnonlin* is in charge of calculating the observation error, which calls *fnUVDErr_C1U* and *fnIMUdltErr* to obtain the camera observation part and the IMU one respectively, while the Jacobian matrix is filled in by $fnJduvd\_CnU_dbg$ and *fnJdaw0_IMU* sequentially.

3.4 The Pre-integration Method

3.4.1 The Observation Vector

For the pre-integration method, the raw camera data are still regarded as observations. However, the raw IMU data are not directly put into the observation vector; instead, they need to be pre-processed to produce some pseudo observations. And the original proposed method [Lupton and Sukkarieh(2012)] is illustrated in Algorithm 1.

Algorithm 1 The Pre-integration Method Based on Inertial Raw Data

```
    $\triangle \mathbf{p}_t^+ = 0$
    $\triangle \mathbf{v}_t = 0$
    $\triangle \mathbf{A}_t = 0$
    for $t_1 < t < t_2$ do
      $\triangle t = t_{t+1} - t_t$
      $\mathbf{f}_t^{bt1} = R_{bt}^{bt1}(\mathbf{f}_t^b - \mathbf{b}_f)$
      $\triangle \mathbf{v}_{t+1} = \triangle \mathbf{v}_t + \mathbf{f}_t^{bt1} \triangle t$
8:    $\triangle \mathbf{p}_{t+1}^+ = \triangle \mathbf{p}_t^+ + \triangle \mathbf{v}_t \triangle t$
      $\triangle \mathbf{A}_{t+1} = \triangle \mathbf{A}_t + E_{bt}^{bt1}(\omega_t^b - \mathbf{b}_\omega) \triangle t$
    end for
    observation = $\begin{bmatrix} \triangle \mathbf{p}_t^+ \\ \triangle \mathbf{v}_t \\ \triangle \mathbf{A}_t \end{bmatrix}$
```

Nonetheless, our simulation results show that this method will produce biased estimation for the velocity of the IMU. After some analysis, we conclude that it is due to the imprecise of $\mathbf{p}_t^+$. Therefore, we propose to substitute the Line 8 in Algorithm 1 with this:

$$\triangle \mathbf{p}_{t+1}^+ = \triangle \mathbf{p}_t^+ + \triangle \mathbf{v}_t \triangle t + \frac{1}{2} \mathbf{f}_t^{bt1} (\triangle t)^2$$

And the final results have verified our ideas.

Accordingly, the produced integration results constitute the new observations for IMU:

$$\begin{aligned} \mathbf{z}_{int} &= (\mathbf{z}_{camera}, \mathbf{z}_{IMUint})' \\ &= (\overbrace{\mathbf{uv}_{11}, \mathbf{uv}_{21}, ..., \mathbf{uv}_{M1}, ..., \mathbf{uv}_{1N}, \mathbf{uv}_{2N}, ..., \mathbf{uv}_{MN}}^{M \times N \times 2}, \\ &\quad \overbrace{d\mathbf{p}_1, d\mathbf{v}_1, d\mathbf{A}_1, d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1) \times 9}) \end{aligned} \tag{41}$$

where $d\mathbf{p}_i = (dx_i, dy_i, dz_i)$, $d\mathbf{v}_i = (dvx_i, dvy_i, dvz_i)$, and $d\mathbf{A}_i = (d\alpha_i, d\beta_i, d\gamma_i)$.

3.4.2 The State Vector $\mathbf{x}$

Similar to the non-pre-integration method, the state vector for the pre-integration method is also made up of the IMU rotation (corresponded Euler angles are represented as $\mathbf{A}_i$)/translation ($\mathbf{T}_i$)/velocity ($\mathbf{v}_i$), feature positions, gravity $\mathbf{g}$ in the initial frame, the relative position between the camera and the IMU (Euler angles corresponding to the relative rotation are represented as $\mathbf{A}_{u2c}$, translation is $\mathbf{T}_{u2c}$), and the biases ($\mathbf{b}_f$ for acceleration and $\mathbf{b}_\omega$ for rotation velocity data). And the biggest difference is that here only the IMU poses corresponding to the key camera frames are listed instead of all the IMU poses at each time step. As a result, the dimension of the problem and therefore its computational cost is reduced significantly.

More specifically, for a system composed of an IMU and a camera navigating with N camera poses and M features, the state vector $\mathbf{x}$ is defined as:

$$\mathbf{x} = (\overbrace{\mathbf{A}_2, \mathbf{T}_2, ..., \mathbf{A}_N, \mathbf{T}_N}^{(N-1)\times 6}, \overbrace{\mathbf{P}_{f1}, ..., \mathbf{P}_{fM}}^{M\times 3}, \overbrace{\mathbf{v}_1, ..., \mathbf{v}_N}^{N\times 3}, \mathbf{g}, \mathbf{A}_{u2c}, \mathbf{T}_{u2c}, \mathbf{b}_f, \mathbf{b}_w) \tag{42}$$

And here $\mathbf{A}_i$s correspond to $\mathbf{A}_{0i}$s in the non-pre-integration method.

3.4.3 The Observation Function $H(\mathbf{x})$

Similar to the non-pre-integration method, for the camera, Formulas 11, 12, 13 and 14 still hold.

On the other hand, the part about the integrated pseudo-observations of IMU in $H(\mathbf{x})$ can be broken into the following three parts (please note that there are typos existing in the paper [Lupton and Sukkarieh(2012)] about the signs of the amending items based on updated estimations of biases $\mathbf{b}_f$ and $\mathbf{b}_\omega$):

$$d\mathbf{p}_i = R_i(\mathbf{T}_{i+1} - \mathbf{T}_i - \mathbf{v}_i\triangle t - \frac{1}{2}\mathbf{g}(\triangle t)^2) - \frac{\partial \triangle \mathbf{p}_t^+}{\partial \mathbf{b}_f}(\mathbf{b}_f - \mathbf{b}_{f0}) - \frac{\partial \triangle \mathbf{p}_t^+}{\partial \mathbf{b}_\omega}(\mathbf{b}_\omega - \mathbf{b}_{\omega 0}) \tag{43}$$

$$d\mathbf{v}_i = R_i(\mathbf{v}_{i+1} - \mathbf{v}_i - \mathbf{g}\triangle t) - \frac{\partial \triangle \mathbf{v}_t}{\partial \mathbf{b}_f}(\mathbf{b}_f - \mathbf{b}_{f0}) - \frac{\partial \triangle \mathbf{v}_t}{\partial \mathbf{b}_\omega}(\mathbf{b}_\omega - \mathbf{b}_{\omega 0}) \tag{44}$$

$$d\mathbf{A}_i = fnABG5R(R_{i+1} * R_i') - \frac{\partial \triangle \mathbf{A}_t}{\partial \mathbf{b}_\omega}(\mathbf{b}_\omega - \mathbf{b}_{\omega 0}) \tag{45}$$

where $i = 2, ..., N$, $fnABG5R$ is a function that can obtain Euler angles from a corresponding rotation matrix, and R_i, E_i correspond to the rotation matrix and rotation rate matrix for the IMU at the time step i. respectively.

Putting all items together, $H(\mathbf{x})$ can be written as:

$$\begin{aligned} H(\mathbf{x}) &= (H_{camera}(\mathbf{x}), H_{IMUint}(\mathbf{x})) \\ &= (\overbrace{u_{11}, v_{11}, ..., u_{M1}, v_{M1}, ..., u_{1N}, v_{1N}, ..., u_{MN}, v_{MN}}^{M\times N\times 2}, \\ &\quad \overbrace{d\mathbf{p}_2, d\mathbf{v}_2, d\mathbf{A}_2, ..., d\mathbf{p}_N, d\mathbf{v}_N, d\mathbf{A}_N}^{(N-1)\times 9}) \end{aligned} \tag{46}$$

3.4.4 Jacobian Matrix

Based on the composition of $H(\mathbf{x})$, the corresponding Jacobian matrix can be calculated.

For camera observations of (u_{ij}, v_{ij}, d_{ij}) which represent the observation of the ith feature at the jth camera pose,

$$\frac{\partial u_{ij}}{\partial \mathbf{P}_{ij}} = [f/z_{ij}, 0, -fx_{ij}/z_{ij}^2] \tag{47}$$

$$\frac{\partial v_{ij}}{\partial \mathbf{P}_{ij}} = [0, f/z_{ij}, -fy_{ij}/z_{ij}^2] \tag{48}$$

$$\frac{\partial d_{ij}}{\partial \mathbf{P}_{ij}} = [0, 0, 1] \tag{49}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{A}_j} = R_{u2c}\frac{\partial R_j}{\partial \mathbf{A}_j}(\mathbf{P}_{i1} - \mathbf{T}_j) \tag{50}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{T}_j} = -R_{u2c}R_j \tag{51}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{A}_{u2c}} = \frac{\partial R_{u2c}}{\partial \mathbf{A}_{u2c}}R_j(\mathbf{P}_{i1} - R_j'\mathbf{T}_{u2c} - \mathbf{T}_j) \tag{52}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{T}_{u2c}} = -R_{u2c} \tag{53}$$

$$\frac{\partial \mathbf{P}_{ij}}{\partial \mathbf{P}_{i1}} = R_{u2c}R_j \tag{54}$$

For $d\mathrm{p}_i$,

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{A}_i} = \frac{\partial R_i}{\partial \mathbf{A}_i}(\mathbf{T}_{i+1} - \mathbf{T}_i - \mathbf{v}_i\triangle t - \frac{1}{2}\mathbf{g}(\triangle t)^2) \tag{55}$$

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{T}_i} = -R_i \tag{56}$$

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{T}_{i+1}} = R_i \tag{57}$$

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{v}_i} = -R_i\triangle t \tag{58}$$

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{g}} = -\frac{1}{2}R_i\triangle t^2 \tag{59}$$

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{b}_f} = -\frac{\partial \triangle \mathbf{p}_t^+}{\partial \mathbf{b}_f} \tag{60}$$

$$\frac{\partial d\mathbf{p}_i}{\partial \mathbf{b}_\omega} = -\frac{\partial \triangle \mathbf{p}_t^+}{\partial \mathbf{b}_\omega} \tag{61}$$

For $d\mathrm{v}_i$,

$$\frac{\partial d\mathbf{v}_i}{\partial R_i} = \frac{\partial R_i}{\partial \mathbf{A}_i}(\mathbf{v}_{i+1} - \mathbf{v}_i - \mathbf{g}\triangle t) \tag{62}$$

$$\frac{\partial d\mathbf{v}_i}{\partial \mathbf{v}_i} = -R_i \tag{63}$$

$$\frac{\partial d\mathbf{v}_i}{\partial \mathbf{v}_{i+1}} = R_i \tag{64}$$

$$\frac{\partial d\mathbf{v}_i}{\partial \mathbf{g}} = -R_i\triangle t \tag{65}$$

$$\frac{\partial d\mathbf{v}_i}{\partial \mathbf{b}_f} = -\frac{\partial \triangle \mathbf{v}_t}{\partial \mathbf{b}_f} \tag{66}$$

$$\frac{\partial d\mathbf{v}_i}{\partial \mathbf{b}_\omega} = -\frac{\partial \triangle \mathbf{v}_t}{\partial \mathbf{b}_\omega} \tag{67}$$

For $d\mathbf{A}_i$,

$$\frac{\partial \triangle \mathbf{A}_i}{\partial \mathbf{A}_i} = \frac{\partial fn}{\partial R} R_{i+1} \frac{\partial R_i}{\partial \mathbf{A}_i} \tag{68}$$

$$\frac{\partial \triangle \mathbf{A}_i}{\partial \mathbf{A}_{i+1}} = \frac{\partial fn}{\partial R} \frac{\partial R_{i+1}}{\partial \mathbf{A}_{i+1}} R_1 \tag{69}$$

$$\frac{\partial \triangle \mathbf{A}_i}{\partial \mathbf{b}_\omega} = -\frac{\partial \triangle \mathbf{A}_t}{\partial \mathbf{b}_\omega} \tag{70}$$

3.4.5 Covariance Matrix

The covariance matrix of the original pre-integration method is obtained through Algorithm 2.

The covariance matrix corresponding to our modified pre-integration method is obtained through Algorithm 3.

3.4.6 The Corresponding Code Functions

In the Matlab code, the raw camera data and IMU data are generated by calling a simulation function *fnSimIMUnFeaturesAtNPoses_helix*, and the noise are added through Line 139 to 162 in the *Main_simuNpose.m*. The state vector x is made up through Line 185 to 250 in this file. Moreover, the observation vector is represented as *Zobs* and is composed from Line 273 to 357 in the same file. On the other hand, *fnCnUPredErr_lsqnonlin* is in charge of calculating the observation error, which calls *fnUVDErr_C1U* and *fnIMUdltErr* to obtain the camera observation part and the IMU one respectively, while the Jacobian matrix is filled in by *fnJduvd_CnU_dbg* and *fnJddpvphi_IMU_dbg* sequentially.

3.5 Adding Pseudo Observations

To faciliate our testing of the problem, especially the observability problem, from time to time, we may want to put additional information into the system. For example, in some cases, assuming known $\mathbf{g}$ value in the initial frame would enable us to estimate the biases ($\mathbf{b}_f$ and $\mathbf{b}_\omega$) without ambiguity even when the designed route is very short and not active enough.

Algorithm 2 The Covariance Matrix for the Pre-integration Method

```
$J_t = \mathbf{I}_{15}$
$R_t = \mathbf{I}_{15}$
**for** $t_1 < t < t_2$ **do**
    $\triangle t = t_{t+1} - t_t$
    $\alpha = \frac{dR_{bt}^{bt1}(\mathbf{f}_t - \mathbf{b}_f)}{d\mathbf{A}_t}$
    $\beta = \frac{dE_{bt}^{bt1}(\omega_t - \mathbf{b}_\omega)}{d\mathbf{A}_t}$
    $F_t = \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3\triangle t & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \alpha\triangle t & -R_{bt}^{bt1}\triangle t & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 + \beta\triangle t & \mathbf{0}_3 & -E_{bt}^{bt1}\triangle t \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$
    $G_t = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ R_{bt}^{bt1}\triangle t & \mathbf{0}_3 \\ \mathbf{0}_3 & E_{bt}^{bt1}\triangle t \\ \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}$
    $J_{t+1} = F_t J_t$
    $R_{t+1}^{+} = F_t R_t F_t' + G_t Q_t G_t'$
**end for**
$J_{t1}^{t2} = J_t$
$R_{t1}^{t2} = R_t$
```

Algorithm 3 The Covariance Matrix for the Pre-integration Method

```
$J_t = \mathbf{I}_{15}$
$R_t = \mathbf{I}_{15}$
**for** $t_1 < t < t_2$ **do**
    $\triangle t = t_{t+1} - t_t$
    $\alpha = \frac{dR_{bt}^{bt1}(\mathbf{f}_t - \mathbf{b}_f)}{d\mathbf{A}_t}$
    $\beta = \frac{dE_{bt}^{bt1}(\omega_t - \mathbf{b}_\omega)}{d\mathbf{A}_t}$
    $F_t = \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3\triangle t & \frac{1}{2}\alpha\triangle t^2 & -\frac{1}{2}R_{bt}^{bt1}\triangle t^2 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \alpha\triangle t & -R_{bt}^{bt1}\triangle t & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 + \beta\triangle t & \mathbf{0}_3 & -E_{bt}^{bt1}\triangle t \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$
    $G_t = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ R_{bt}^{bt1}\triangle t & \mathbf{0}_3 \\ \mathbf{0}_3 & E_{bt}^{bt1}\triangle t \\ \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix}$
    $J_{t+1} = F_t J_t$
    $R_{t+1}^{+} = F_t R_t F_t' + G_t Q_t G_t'$
**end for**
$J_{t1}^{t2} = J_t$
$R_{t1}^{t2} = R_t$
```

For this purpose, we have made several options available including $\mathbf{g}$, $\mathbf{A}_{u2c}$, $\mathbf{T}_{u2c}$, $\mathbf{b}_f$ and $\mathbf{b}_\omega$. And the users can arbitrarily choose the combination of the pseudo observations by assigning their corresponding bool variables ($bAddZg$ for $\mathbf{g}$, $bAddZau2c$ for $\mathbf{A}_{u2c}$, $bAddZtu2c$ for $\mathbf{T}_{u2c}$, $bAddZbf$ for $\mathbf{b}_f$ and $bAddZbw$ for $\mathbf{b}_\omega$) to 1s. Once enabled, we assign high weight to their corresponding information matrix items ($1e8$) to ensure the imposed information is effective. And their corresponding blocks in the Jacobian matrix are all assigned to identity matrices.

4 The Computation of Initial Values

When faced with a real scenario, the first issue we need to deal with after formulating the problem is obtaining the initial values of the chosen state vector. Given $dp_i, dv_i, d\phi_i$, $\mathbf{g}$ and $\mathbf{v}_i$ can be obtained as follows (please note that there are typos about the related formulas in the original paper):

$$\begin{aligned} \mathbf{T}_{i+1} &= \mathbf{T}_i + (t_{i+1} - t_i)\mathbf{v}_i + R'_i \triangle p_i^+ + \frac{1}{2}(t_{i+1} - t_i)^2 \mathbf{g} \\ \Longrightarrow \mathbf{v}_i &= \frac{\mathbf{T}_{i+1} - \mathbf{T}_i - R'_i \triangle p_i^+ - \frac{1}{2}(t_{i+1} - t_i)^2 \mathbf{g}}{t_{i+1} - t_i} \end{aligned} \tag{71}$$

$$\left.\begin{aligned} \mathbf{v}_i &= \frac{\mathbf{T}_{i+1} - \mathbf{T}_i - R'_i \triangle p_i^+}{t_{i+1} - t_i} - \frac{1}{2}(t_{i+1} - t_i)\mathbf{g} \\ \mathbf{T}_{i+1} &= \mathbf{T}_i + (t_{i+1} - t_i)\mathbf{v}_i + R'_i \triangle p_i^+ + \frac{1}{2}(t_{i+1} - t_i)^2 \mathbf{g} \\ \mathbf{T}_{i+2} &= \mathbf{T}_{i+1} + (t_{i+2} - t_{i+1})\mathbf{v}_{i+1} + R'_{i+1} \triangle p_{i+1}^+ + \frac{1}{2}(t_{i+2} - t_{i+1})^2 \mathbf{g} \\ \mathbf{v}_{i+1} &= \mathbf{v}_i + (t_{i+1} - t_i)\mathbf{g} + R'_i \triangle v_i \end{aligned}\right\} \tag{72}$$

$$\Longrightarrow \mathbf{T}_{i+2} = \mathbf{T}_{i+1} + (t_{i+2} - t_{i+1})(\mathbf{v}_i + (t_{i+1} - t_i)\mathbf{g} + R'_i \triangle v_i) + R'_{i+1} \triangle p_{i+1}^+ + \frac{1}{2}(t_{i+2} - t_{i+1})^2 \mathbf{g} \tag{73}$$

$$= \mathbf{T}_{i+1} + (t_{i+2} - t_{i+1})(\frac{\mathbf{T}_{i+1} - \mathbf{T}_i - R'_i \triangle p_i^+}{t_{i+1} - t_i} + \frac{1}{2}(t_{i+1} - t_i)\mathbf{g} + R'_i \triangle v_i) + R'_{i+1} \triangle p_{i+1}^+ + \frac{1}{2}(t_{i+2} - t_{i+1})^2 \mathbf{g} \tag{74}$$

$$\Longrightarrow \mathbf{g} = \frac{\mathbf{T}_{i+2} - \mathbf{T}_{i+1} - (t_{i+2} - t_{i+1})(\frac{\mathbf{T}_{i+1} - \mathbf{T}_i - R'_i \triangle p_i^+}{t_{i+1} - t_i} + R'_i \triangle v_i) - R'_{i+1} \triangle p_{i+1}^+}{\frac{1}{2}(t_{i+2} - t_{i+1})(t_{i+2} - t_i)} \tag{75}$$

5 Bibliography

References

[Lupton and Sukkarieh(2012)] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012. ISSN 15523098. doi: 10.1109/TRO.2011.2170332.