Robotic Embedded Systems Laboratory
Technical Note TN-2010-03

Department of Computer Science
University of Southern California
http://robotics.usc.edu/resl/

A User's Guide to the Camera-IMU Calibration Toolbox for MATLAB

Jonathan Kelly

September 8, 2011

1 Introduction

This document is a brief user's guide for the Camera-IMU Calibration Toolbox for MATLAB, a.k.a. Invisical (an acronym for INertial-VISual Integrated CALibration). The toolbox can be used to accurately determine the 6-DOF rigid body transform between a camera and an inertial measurement unit (cf. Figure 1).

In the sections that follow, we review toolbox installation, setup, and use. If you have additional questions, comments, or bugs to report, please email Jonathan Kelly (`jonathsk@usc.edu`).

2 Installation

Installation is performed in three easy steps:

1. copy the `invisical/` directory and its contents to a convenient, MATLAB-accessible location,
2. `cd` to the (new) `invisical/` directory from within MATLAB, and
3. run the `setpath` script to update the MATLAB path.

Optionally, to avoid having to run `setpath` again in the future, the `savepath` command can be used to save the updated path information to the MATLAB `pathdef.m` file (which is read on startup).

3 Preparing to Calibrate: Required Files

The calibration algorithm requires several inputs beyond raw IMU and camera data. For example, we need models of the individual camera and IMU sensors, an initial estimate of the relative pose of the sensors etc. We describe each required input file below.

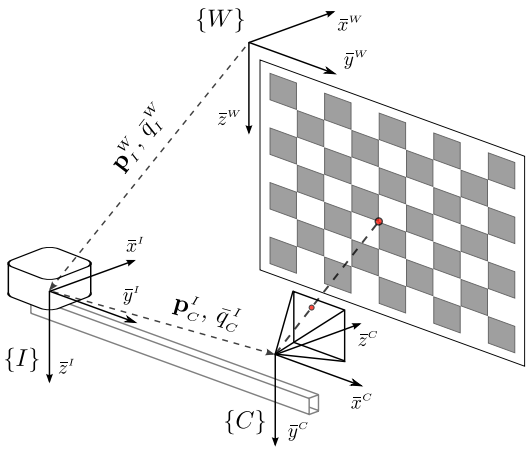

Figure 1: Relationship between the world $\{W\}$, IMU $\{I\}$, and camera $\{C\}$ reference frames. The IMU (left) and camera (right) are rigidly attached to a common platform. The goal of calibration is to determine the transform $(\mathbf{p}_C^I, \bar{q}_C^I)$ between the camera and IMU.

3.1 MATLAB Calibration Setup File

The calibration setup file is a MATLAB `.m` file that implements a single setup function, which returns a structure containing a series of calibration settings and algorithm parameters. The contents of a setup file include:

```
function [setup] = example_setup

setup.name = 'Camera-IMU Calibration Setup - Example File';
setup.desc = 'Description goes here, if desired.';

% Files and directories.
baseDir = '/home/jonathan/Academic/Research/invisical';

setup.data.poseFile = fullfile(pwd, 'setup/mrp_pose_001.mat');
setup.data.imuFile = fullfile(baseDir, 'fullcal-1/fullcal-1.imu');
setup.data.ptsDir = fullfile(baseDir, 'fullcal-1/points');
setup.data.imgDir = fullfile(baseDir, 'fullcal-1/rectified');
...
```

```
...
setup.files.cmodFile = fullfile(baseDir, 'camcal/model_left.xml');
setup.files.imodFile = fullfile(baseDir, 'imucal/3DM-GX3.xml');
setup.files.gmodFile = fullfile(baseDir, 'imucal/gravity_LA.xml');

% Timing and stepping.
setup.data.camDelay = 0.049;
setup.data.camSteps = 1;
setup.data.firstImg = 745;
setup.data.finalImg = 1600;

% Iterated filtering.
setup.ispkf.maxIters = 25;
setup.ispkf.deltaTol = 0.001;
```

Stepping through the file line-by-line:

- `function [setup] = example_setup`

  The function returns a single structure, `setup`, which contains several sub-structures.

- `setup.name = 'Camera-IMU Calibration Setup - Example File';`

  Each setup file should be identified by a descriptive name - it is useful in some situations to maintain several setup files for one calibration data set.

- `setup.desc = 'Description goes here, if desired.';`

  An optional text description can also be included, if desired.

- `baseDir = '/home/jonathan/Academic/Research/invisical';`

  Here, we use the `baseDir` variable within the function itself to represent the base directory for the various calibration files - this avoids repeating the same string several times.

- `setup.data.poseFile = fullfile(pwd, 'setup/mrp_pose_001.mat');`

  Pointer to the initial pose estimate file. See Section 3.5.

- `setup.data.imuFile = fullfile(baseDir, 'fullcal-1/fullcal-1.imu');`

  Pointer to the IMU calibration data file, an ASCII file containing timestamped IMU measurements. See Section 4.1.

- `setup.data.ptsDir = fullfile(baseDir, 'fullcal-1/points');`

  Pointer to the directory containing processed camera data files. See Section 4.2.

- `setup.data.imgDir = fullfile(baseDir, 'fullcal-1/rectified');`

  Pointer to the directory containing rectified camera image files. These files are not used directly for calibration. Invisical is able to display the predicted and measured locations of feature points during calibration, for verification/debugging.

- `setup.files.cmodFile = fullfile(baseDir, 'camcal/model_left.xml');`

  Pointer to a camera model XML file. See Section 3.2.

- `setup.files.imodFile = fullfile(baseDir, 'imucal/3DM-GX3.xml');`

  Pointer to an IMU model XML file. See Section 3.3.

- `setup.files.gmodFile = fullfile(baseDir, 'imucal/gravity_LA.xml');`

  Pointer to a gravity model XML file. See Section 3.4.

- `setup.data.camDelay = 0.049;`

  There is often a latency or delay between the camera and IMU data streams, due to the operating system, logging hardware etc. The `camDelay` value represents a constant time shift (latency), in seconds, for all of the image timestamps relative to the IMU timestamps - when the camera data is loaded, this value is *subtracted* from each image timestamp. For a positive delay, this effectively moves the camera measurements backwards in time relative to the IMU measurements.

- `setup.data.camSteps = 1;`

  Normally, the calibration algorithm loads and processes each camera data file in sequence. When images have been captured a high frame rate, this may not be necessary. The `camSteps` parameter controls the image step size, i.e., if the parameter is set to 3, only every third camera data file is processed etc.

- `setup.data.firstImg = 745;`

  Index of the first camera data file to process for calibration (according to timestamp).

- `setup.data.finalImg = 1600;`

  Index of the final camera data file to process for calibration (according to timestamp).

- `setup.ispkf.maxIters = 25;`

  In addition to the standard unscented Kalman filter, Invisical supports the iterated unscented Kalman filter, which is also called the iterated sigma point Kalman filter (ISPKF). When supplied, this field determines the maximum number of iterated measurement updates to perform when processing each camera data file. Note that this can significantly increases the overall calibration time. Comment out this line to use the UKF only (equivalent to setting `maxIters = 1`).

- `setup.ispkf.deltaTol = 0.001;`

  Convergence threshold for sequential measurement updates in the ISPKF. Presently, the algorithm computes the difference (delta) between sequential updates of the first seven states in the state vector. These correspond to the IMU position and orientation in the world frame; if the 2-norm of the difference is less than `deltaTol`, the update is considered to have converged, and processing continues with the next propagation step.

Note that the locations of the IMU model, camera model, and gravity model files can also be provided directly as arguments to the Invisical setup routine. These arguments will override any locations specified in the setup file.

3.2 IMU Model

The filtering algorithm employed by Invisical treats IMU measurements as control inputs in the system dynamics equations. To propagate the uncertainty associated with the state estimate forward in time, we need to know the noise characteristics of the IMU. These values are suppled in an XML IMU model file - the model file for the MicroStrain 3DM-GX3-25 is included here as an example.

```
<?xml version="1.0" encoding="utf-8"?>
<imu>
  <model> IMU_3AXIS_BASIC </model>
  <noise>
    <gyroscope>
      <xstdev> 0.001112 </xstdev>
      <ystdev> 0.001112 </ystdev>
      <zstdev> 0.001008 </zstdev>
    </gyroscope>
    <accelerometer>
      <xstdev> 0.001564 </xstdev>
      <ystdev> 0.001525 </ystdev>
      <zstdev> 0.002169 </zstdev>
    </accelerometer>
  </noise>
  <drift>
    <gyroscope>
      <xstdev> 0.000226 </xstdev>
      <ystdev> 0.000268 </ystdev>
      <zstdev> 0.000231 </zstdev>
    </gyroscope>
    <accelerometer>
      <xstdev> 0.000482 </xstdev>
      <ystdev> 0.000588 </ystdev>
      <zstdev> 0.001065 </zstdev>
    </accelerometer>
  </drift>
</imu>
```

The `<noise>` section of the file defines the standard deviation of the noise (i.e., the angle random walk values) for each gyroscope and accelerometer axis. Similarly, the `<drift>` section of the file defines the sensor drift (i.e., the bias stability values).

3.3 Camera Model

Invisical uses a linear camera model, defined in an XML file, to project landmarks in the world frame onto the camera image plane. The model is a standard linear Tsai (Brown) model [1] - we assume that the images have already been rectified to remove lens distortions.

```
<?xml version="1.0" encoding="utf-8"?>
<camera>
  <model> CAMERA_LINEAR </model>
  <imgsize>
    <width> 640.000000 </width>
    <height> 480.000000 </height>
  </imgsize>
  <intrinsic>
    <fx> 566.636976 </fx>
    <fy> 565.986869 </fy>
    <sx> 0.000000 </sx>
    <cx> 336.591741 </cx>
    <cy> 226.714768 </cy>
  </intrinsic>
</camera>
```

The `<imgsize>` tags, `<width>` and `<height>`, define the size of the image plane, in pixels, while the `<intrinsic>` tags define the individual elements of the (linear) camera intrinsic calibration matrix $\mathbf{K}$,

$$\mathbf{K} = \begin{bmatrix} f_x & s_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where f_x and f_y are the lens x and y focal lengths, respectively, c_x and c_y define the principal point, and s_x is the image skew (usually zero).

3.4 Gravity Model

The calibration algorithm requires an initial estimate of the gravity vector in the world reference frame. This initial estimate is defined by a gravity model file, which is a simple XML file with the syntax:

```
<?xml version="1.0" encoding="utf-8"?>
<gravity>
  <model> GRAVITY_SIMPLE </model>
  <vector>
    <f> 0.000000 </f>
    <r> 0.000000 </r>
    <d> 9.796000 </d>
  </vector>
</gravity>
```

The file identifies the type of model (`GRAVITY_SIMPLE` in this case), and gives the forward (x), right (y) and down (z) components of the gravity vector, in units of m/s^2.

3.5 Camera-IMU Relative Pose Estimate

We also require an initial, rough estimate of the camera-IMU transform, and its associated uncertainty. This estimate can be obtained from hand measurements, available CAD models, or other sources. The transform parameters are supplied to Invisical as a `.mat` file that contains a single MATLAB structure, `relPose`, with three fields:

```
>> load mrp_pose.mat;
>> relPose
relPose =
    pic:  [3x1 double]
    qic:  [4x1 double]
    Sic:  [6x6 double]
```

The `pic` field defines the relative translation, $\mathbf{p}_C^I$, of the optical center of the camera in the IMU frame (in units of m). Likewise, the `qic` field defines the (unit quaternion) orientation, $\bar{q}_C^I$, of the camera frame with respect to the IMU frame. Our convention, used throughout Invisical, is to specify the scalar component of the quaternion $\bar{q}$ as `q(1)`.

The remaining field, `Sic`, contains a 6×6 covariance matrix for the pose estimate. This covariance matrix will typically, but not always, be diagonal. The upper 3×3 submatrix defines the translation uncertainty, while the lower 3×3 matrix defines the orientation uncertainty. For the orientation uncertainty, we use a modified Rodriguez parameters error state representation (which is a minimal representation). For more information on the modified Rodriguez parameters, see related TechNote TN-2009-02.

3.6 Other Parameters of Interest

In addition to the setup and model files, there are two other sets of parameters that may need to be adjusted:

- `imuParams.gravityUncert` (`invis_setup_quat.m`, line 76)

  Defines the initial uncertainty (standard deviation) of the gravity vector in the world frame.

- `camParams.pixelNoise` (`invis_setup_quat.m`, line 98)

  Defines the standard deviation of the camera measurement (feature extraction) noise.

At present, these parameter values are hard-coded in `invis_setup_quat.m`, however this will change in a future release.

4 Preparing to Calibrate: Data Collection and Preprocessing

Data collection typically involves moving the camera-IMU platform in front of a planar calibration target, such that the target remains approximately within the camera's field of view. Under normal circumstances, between one and two minutes of data are required for calibration (at 7.5 or 15 camera frames per second).

The calibration algorithm assumes that the camera-IMU platform starts from rest, and that the camera has an approximately fronto-parallel configuration relative to the planar target initially. During data collection, it is

important to excite the full six degrees of rotational and translational freedom in the system, to ensure that all of the calibration parameters are observable.

Once data collection is complete, the IMU and camera measurements must be pre-processed according to the specifications below.

4.1 IMU Data

IMU measurements are stored in an ASCII text file, ordered sequentially according to time. Each line of the file has seven fields, separated by whitespace, in the following order: `<UNIX timestamp> <accel. x> <accel. y> <accel. z> <gyro. x> <gyro. y> <gyro. z>`. Several lines from an example IMU data file for the MicroStrain 3DM-GX3-25 unit are shown below.

```
1249898577.875835 0.041180 -0.009030 -0.999191 0.009645 0.002427 -0.006585
1249898577.883836 0.041668 -0.009519 -0.999557 0.007911 0.001744 -0.005048
1249898577.895848 0.043128 -0.009653 -0.999176 0.013034 0.005511 -0.000302
...
```

Note that, in the case of the 3DM-GX3-25, the accelerometer measurements are reported in units of g's, and are converted to SI units by the Invisical software prior to the start of the calibration routine.

4.2 Camera Data

While all of the IMU measurements are grouped together in a monolithic file, the measurements for each camera image are stored individually, with one `.mat` file per frame. The image data files contain feature correspondence information only - we assume that image rectification and feature extraction/matching have already been performed. An image data file contains a single MATLAB structure, called `data`, with the fields listed in the example below.

```
>> load 1-0000215.mat;
>> data
data =
     file:  '1-000215_rect.pgm'
   tstamp:  1.2499e+09
    world:  [3x48 double]
    image:  [2x48 double]
    valid:  1
```

The `file` field identifies the original (rectified) image file; the original image is not used during calibration, but may be used for visualization, if desired. Each image has an associated timestamp, which is stored in the `tstamp` field (UNIX timestamp format, μs resolution).

The `world` field contains a $3 \times n$ array of (known) 3D world points (e.g., corners on the calibration target), while the `image` field contains a $2 \times n$ array of the corresponding image plane projections (i.e., detected feature points).

Finally, the `valid` field is a Boolean flag that indicates whether or not the data file should be processed. In some cases, it may be useful to suppress certain images, due to, e.g., noise, inaccurate feature extraction etc. The calibration algorithm will ignore any data files with the `valid` field set to `false` or `0`.

5 Running the Calibration Algorithm

Once the setup and model files are available in known locations, and the data has been collected and pre-processed, it is possible to run the calibration algorithm. Calibration can be initiated from the command prompt (Section 5.1) or from a GUI window (Section 5.2).

5.1 From the MATLAB Command Prompt

To run calibration from the MATLAB command prompt, it is necessary to first load the calibration setup parameters (line 1 below), and then to call the `invis_setup_quat` function to load the calibration data and initialize the state vector.

```
>> setup = example_setup;
>>[params, dataset, control] = invis_setup_quat(setup);
[invis_setup_quat]:  Loaded IMU data.  Total of 24174 measurements.
[invis_setup_quat]:  Loaded cam data.  Total of 3410 images.
[invis_setup_quat]:  First image is 4.53 seconds after IMU start.

[invis_setup_quat]:  Processing start: 53.89 s, End: 111.14 s, Dur: 57.25 s
[invis_setup_quat]:  Using 5726 IMU measurements and 856 images to calibrate.

[invis_setup_quat]:  Estimated gyro. biases: 0.001,  0.005, -0.002
[invis_setup_quat]:  Estimated accl. biases: 0.380, -0.053, -0.009

[invis_setup_quat]:  Estimated cam position:  [ -2.76, 0.27, 0.35]
[invis_setup_quat]:  Estimated IMU position:  [ -2.76, 0.42, 0.34]
```

The `invis_setup_quat` function returns three structures: `params`, which contains a variety of estimation parameters, `dataset`, which contains the camera and IMU data sets, and `control`, which contains a series of algorithm control flags (visualization, interactivity). These structures are then passed (after any manual adjustments) to the `invis_calib_seq` sequential calibration function.

```
>> output = invis_calib_seq(params, dataset, control)
State vector at time t = 53.88824
...

State vector at time t = 54.12893
...

[invis_calib_seq]:  Processing image l-000744_rect.pgm
[invis_calib_seq]:  Step time was 4.16 sec.  Estimate 1.32 hours remain.
State vector at time t = 54.12893

pwi:  -2.76341   qwi:  0.02241   vwi:  0.05930
       0.41992         0.01201        -0.09765
       0.34315        -0.01507         0.01625
                      -0.99956

 bg:   0.00066    ba:  0.38022    gw: -0.00195
       0.00513        -0.05278         0.00008
      -0.00154        -0.00907         9.79608

pic:   0.00027   qic:  0.50473
       0.14906         0.49025
       0.00138        -0.49831
                      -0.50654
```

5.2 From the GUI Panel

It is also possible to run the calibration algorithm from a MATLAB GUI window. To launch the GUI, type `invis_gui` at the MATLAB command prompt. The control window shown in Figure 2 should appear; each of the seven data sources (setup file, IMU model, camera model, gravity model, relative pose file, IMU data file, camera data files) must be loaded before the 'Calibrate' button will enable. At the present time, the GUI components are alpha software – your mileage may vary.

References

[1] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, 1st ed., ser. Interdisciplinary Applied Mathematics. New York: Springer, November 2004, vol. 26.

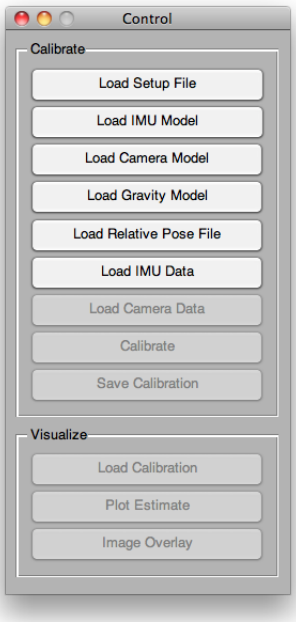

Figure 2: MATLAB Invisical GUI application window (alpha).