Moving a MATLAB 3D point cloud registration algorithm into a smartphone and Bluetooth iBeacon environment

Dr R Bruce Backman
Australian Defence Science & Technology Group,
Sensor Processing & Algorithms
bruce.backman@dst.defence.gov.au
+61 419 741 008
Motivation for this problem

Introducing a test algorithm

How Matlab currently cohabits in the cloud, desktop computer, iOS smartphone and Bluetooth beacon ecosystem

Test algorithm written in Matlab

Test algorithm converted in Matlab coder app

Test algorithm running in target environment (macos)

Lessons learnt

Conclusions and recommendations
**Motivation for this problem**

- Sensor Processing and Algorithms Group requested to produce 3D point cloud manipulation support by other project groups
- SPA has been working closely with Estimote Beacons to guide the development of a Beacon which is more suited to a GPS-denied, dynamic environment
- *Most critical need* is to find a way to run Matlab algorithms on Beacon-based dynamic environment away from desktop computing
- The next wave of computing will be based in a physical context defined by ubiquitous beacons. Our current model of a specialist writing highly specialized Matlab code on a personal computer will have to change
INTRODUCING A TEST ALGORITHM
Anatomic Fiducials – With Injury Region of Interest
SVD-based 3D Registration Algorithm

Two 3-D point sets \( \{p_i\}\) and \( \{p'_i\}\); \( i = 1, 2 \ldots, n \), represented as 3 × 1 column matrices

\[
p' = \frac{1}{n} \sum_{i=1}^{n} p'_i, \quad p'' = \frac{1}{n} \sum_{i=1}^{n} p'' = \hat{R}p + \hat{T}, \quad p = \frac{1}{n} \sum_{i=1}^{n} p_i
\]

\[
q_i = p_i - p \quad \text{and} \quad q'_i = p'_i - p'
\]

\[
\Sigma^2 = \sum_{i=1}^{n} \|q'_i - Rq_i\|^2
\]

\[
\hat{T} = p' - \hat{R}p.
\]

\[
H = \sum_{i=1}^{N} q_i q'_i^T
\]

\[
SVD(H) = UVU^T
\]

If \( \det(VU^T) = 1 \), then \( \hat{R} = VU^T \), If \( \det(VU^T) = -1 \), no solution better than a reflection

\[
p'_i = Rp_i + T + N_i
\]
Dimensionality of an Embedded Spherical Panorama, or *Image Bubble*

A 2D representation of the interior of the spherical panorama (Image Bubble) where ‘+’ represents the origin.

N.B. If this is a 2D slice representing a 3D sphere, which is spatially anchored (embedded) within the Terrain Cube then the Terrain Cube, by inference, must be one dimension higher, or 4D.
How Matlab currently cohabits in the cloud, desktop computer, iOS smartphone and Bluetooth beacon ecosystem
Test algorithm written in Matlab
function aruns_algorithm()
% Sensor Processing & Algorithms Group - Shawn Garner, Group Leader
% Author: R Bruce Backman
verbose = true;
singular_threshold = 100;

% Construct 'H' matrix by processing 3D point cloud subsets
TerrainDataset1 = [...
43.5000 -704.5200 739.3600;
67.3200 -631.5200 726.1800;
73.4800 -608.0400 719.6400;
69.2200 -585.2200 723.2600;
40.1000 -514.3200 730.9800;
117.7200 -611.4600 658.6200;
148.9000 -613.1800 583.2400;
];

TerrainDataset2 = [...

Command Window
New to MATLAB? See resources for Getting Started.

***************---------------------------***************

X is a rotation with the desired properties, i.e. that is, not a reflection.
Information: Condition number of W indicates a well-conditioned matrix.
I.e. the mathematics foundations assumed prior to using this technique have been met.
Integrity check shows algorithm results correct

***************---------------------------***************

fX >>
%% Find R' & T' & define N'
% Rotation matrix to bring p set 3D points into alignment with p' 3D points
Rhat = V_transpose * U';
% Translation vector to shift the centroids of the two 3D sets together
That = p_prime_centroid - Rhat * p_centroid;
% A 'Noise' vector to compensate for breakdowns in the SVD process... not
% used in my experience, with 3D points from Ladar sensors, etc. RBB
Nhat = zeros(size(That));

% Algorithmic success check #1
% Test the singular values on the diagonal of W
% I.e. test value (a.k.a 'condition number' is equal to max(diag(W))/min(diag(W))
if isnan((max(diag(W))/min(diag(W))))
    if verbose,fprintf('Warning: Condition number of W is infinite...
' );end
elseif (max(diag(W))/min(diag(W)) >= singular_threshold) && verbose
    if verbose,fprintf('Information: Condition number of W indicates an ill-conditioned matrix
');end
else
    if verbose
        fprintf('Information: Condition number of W indicates a well-conditioned matrix.
');
        fprintf('I.e. the mathematics foundations assumed prior to using this technique have been met.
');
    end
end

% Algorithm Check: back calculating to give p'
if mean(Rhat*p_centroid+That+Nhat-p_prime_centroid(:))<eps
    if verbose,fprintf('Integrity check shows algorithm results correct
');end
end
fprintf('
*****************************

X is a rotation with the desired properties, i.e. that is, not a reflection.
Information: Condition number of W indicates a well-conditioned matrix.
I.e. the mathematics foundations assumed prior to using this technique have been met.
Integrity check shows algorithm results correct

*****************************

fx >>
% Sensor Processing & Algorithms Group - Shawn Garner, Group Leader
% Author: R Bruce Backman
% Purpose of code: To drive: => to demonstrate a technique for merging two 3D point
% cloud data sets based on a smaller subset of each where the subset points
% represent the same fiducials or landmarks in the larger point clouds.
%
% Note that this function has been written to be translated into C code
% by the Matlab Coder app and may not be as elegant as normal.

aruns_algorithm()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X is a rotation with the desired properties, i.e. that is, not a reflection.
Information: Condition number of W indicates a well-conditioned matrix.
I.e. the mathematics foundations assumed prior to using this technique have been met.
Integrity check shows algorithm results correct.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Test algorithm converted in Matlab Coder app
plot_3D_points(q, q_prime);

H = q' * q_prime; % H is now given to SVD for decomposition

% Note, for C/C++ translation purposes the 'H' matrix is cast to a Real
% matrix, with all 01's, so that the three output matrices are cast to
% Reals with no mistakes -- RBB, (technical note for compiler level.)
[U, W, V_transpose] = svd(real(H));
X = ((V_transpose')') * U';

fprintf('------------------------------

if (abs(1-det(X)) < eps)
    if verbose,fprintf('
X is a rotation with the desired properties, i.e. that is, not a reflection.
');end
else (-1-(det(X)) > eps)
    if verbose,fprintf('
Caution: rotation solution no better than a reflection through a plane. 
');end
else
    if verbose,fprintf('
Warning: Proceed with caution as determinant(X) test pathological.
');end

% Find R' & T' & define N'
% Rotation matrix to bring p set 3D points into alignment with p' 3D points
Rhat = V_transpose * U';

% Translation vector to shift the centroids of the two 3D sets together
That = p_prime_centroid - Rhat * p_centroid;

% A 'Noise' vector to compensate for breakdowns in the SVD process... not
% used in my experience, with 3D points from Ladar sensors, etc. RBB
Nhat = zeros(size(That));

% Algorithmic success check #1
% Test the singular values on the diagonal of W
% I.e. test value (a.k.a 'condition number' is equal to max(diag(W))/min(diag(W))
if isinf((max(diag(W))/min(diag(W))))
    if verbose,fprintf('Warning: Conondition number of W is infinite...
');end
else (max(diag(W))/min(diag(W)) >= singular_threshold) && verbose
    if verbose,fprintf('Information: Condition number of W indicates an ill-conditioned matrix
');end
else
    if verbose
        fprintf('Information: Condition number of W indicates a well-conditioned matrix.
');
        fprintf('I.e. the mathematics foundations assumed prior to using this technique have been met.
');
    end
end
Test algorithm running in target environment (MacOS)
```c
/* Include files */
#include "aruns_algorithm.h"
#include "main.h"

/* Function Declarations */
static void main_aruns_algorithm();

/* Function Definitions */
static void main_aruns_algorithm()
{
    /* Call the entry-point 'aruns_algorithm'. */
    aruns_algorithm();
}

int main(int, const char * const [])
{
    /* Initialize the application. */
    You do not need to do this more than one time. */
    /* Invoke the entry-point functions. */
    You can call entry-point functions multiple times. */
    main_aruns_algorithm();

    /* Terminate the application. */
    You do not need to do this more than one time. */
    aruns_algorithm.terminate();
    return 0;
}
```
```c
void aruns_algorithm()
{
    int i0;
    double q[21];
    static const double cv0[7] = { 17.386666666666666666, 41.286666666666666666, 47.366666666666666666, 63.106666666666666666, 13.986666666666666666, 91.086666666666666666, 122.786666666666666666 };;
    static const double cv1[7] = { -758.616666666666666666, -605.616666666666666666, -642.803333333333333333, -649.216666666666666666, -604.383333333333333333, -646.466666666666666666, -677.176666666666666666 };;
    static const double cv2[7] = { 677.666666666666666666, 654.486666666666666666, 657.946666666666666666, 661.566666666666666666, 669.286666666666666666, 599.926666666666666666, 521.546666666666666666 };;
    double q_prime[21];
    static const double cv3[7] = { -0.1803999999999998, -0.0599999999999999, -0.0449999999999999, -0.0251999999999999, -0.0170999999999999, -0.1010999999999999, -0.0510999999999999 };;
    static const double cv4[7] = { 0.0462333333333333, -0.0199666666666666, -0.0492666666666666, -0.0326666666666666, -0.0248666666666666, -0.0156666666666666, -0.0051666666666666 };;
    static const double cv5[7] = { 0.0825666666666666, 0.8739666666666666, 0.8641666666666666, 0.8693666666666666, 0.8762666666666666, 0.8045666666666666, -0.0706333333333333 };;
    double b_c[9];
    creal_T b_U[3];
    double s[3];
    creal_T V[3];
    int i0;
    double W[3];
    double V_ro;
    int k;
    creal_T X[3];
    creal_T dc0;
    double V_im;
    double U_ro;
    double U_im;
    creal_T dcv0[3];
    static const double cv0[3] = { 26.11333333333334, 53.99333333333334, 0.0933333333333334 };;
    creal_T x[3];
    creal_T b_X[3];
    creal_T dev1[3];
    creal_T dcv1[3];
    static const double cv1[3] = { 0.1989999999999998, 0.1788666666666667, 0.1746333333333333 };;
    creal_T y[3];
    double a[3];
    double q_prime[3];
    double s[3];
    creal_T V[3];
    int i0;
    double W[3];
    double V_ro;
    int k;
    creal_T X[3];
    static const double cv[3] = { 26.11333333333334, 53.99333333333334, 0.0933333333333334 };;
    creal_T x[3];
    creal_T b_X[3];
    creal_T dev1[3];
    creal_T dcv1[3];
    static const double cv1[3] = { 0.1989999999999998, 0.1788666666666667, 0.1746333333333333 };;
    creal_T y[3];
    double a[3];
    double q_prime[3];
    double s[3];
    creal_T V[3];
    int i0;
    double W[3];
    double V_ro;
    int k;
    creal_T X[3];
    static const double cv[3] = { 26.11333333333334, 53.99333333333334, 0.0933333333333334 };;
    creal_T x[3];
    creal_T b_X[3];
    creal_T dev1[3];
    creal_T dcv1[3];
    static const double cv1[3] = { 0.1989999999999998, 0.1788666666666667, 0.1746333333333333 };;
    creal_T y[3];
    double a[3];
    double q_prime[3];
    double s[3];
    creal_T V[3];
    int i0;
    double W[3];
    double V_ro;
    int k;
    creal_T X[3];
    static const double cv[3] = { 26.11333333333334, 53.99333333333334, 0.0933333333333334 };;
    creal_T x[3];
    creal_T b_X[3];
    creal_T dev1[3];
    creal_T dcv1[3];
    static const double cv1[3] = { 0.1989999999999998, 0.1788666666666667, 0.1746333333333333 };;
    creal_T y[3];
    double a[3];
    double q_prime[3];
    double s[3];
    creal_T V[3];
    int i0;
    double W[3];
    double V_ro;
    int k;
    creal_T X[3];
    static const double cv[3] = { 26.11333333333334, 53.99333333333334, 0.0933333333333334 };;
    creal_T x[3];
    creal_T b_X[3];
    creal_T dev1[3];
    creal_T dcv1[3];
    static const double cv1[3] = { 0.1989999999999998, 0.1788666666666667, 0.1746333333333333 };;
    creal_T y[3];
    double a[3];
    double q_prime[3];
    double s[3];
    creal_T V[3];
    int i0;
    double W[3];
    double V_ro;
    int k;
This file initializes entry-point function arguments to a default.

- size and value before calling the entry-point functions. It does
- not store or use any values returned from the entry-point functions.
- If necessary, it does pre-allocate memory for returned values.
- You can use this file as a starting point for a main function that
  you can deploy in your application.

- After you copy the file, and before you deploy it, you must make the
  following changes:
  - For variable-size function arguments, change the example sizes to
  - the sizes that your application requires.
  - Change the example values of function arguments to the values that
  - your application requires.
  - If the entry-point functions return values, store these values or
  - otherwise use them as required by your application.

- Include files

### Include
- aruns_algorithm.h
- main.h

### Function Declaration

```c
static void main_aruns_algorithm();
```

### Function Definition

```c
static void main_aruns_algorithm()
{
    /* Call the entry-point 'aruns_algorithm'. */
    aruns_algorithm();
}
```

```c
int main(int, const char * const [])
{
    /* Initialize the application.
    You do not need to do this more than one time. */
```

---

X is a rotation with the desired properties, i.e. that is, not a reflection.
Information: Condition number of W indicates a well-conditioned matrix.
I.e. the mathematics foundations assumed prior to using this technique have been met.
Integrity check shows algorithm results correct

Program ended with exit code: 0
Last login: Thu Apr 26 09:30:13 on ttys003
Dr-R-Bruce-Backmans-Mac-Pro-741:~ backmanb$ /Users/backmanb/Aruns_Algorithm_V3 ; exit;

X is a rotation with the desired properties, i.e. that is, not a reflection.
Information: Condition number of W indicates a well-conditioned matrix.
I.e. the mathematics foundations assumed prior to using this technique have been met.
Integrity check shows algorithm results correct

logout
Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]

X is a rotation with the desired properties, i.e. that is, not a reflection.
Information: Condition number of W indicates a well-conditioned matrix.
I.e. the mathematics foundations assumed prior to using this technique have been met.
Integrity check shows algorithm results correct

Program ended with exit code: 0
Lessons learnt

- Learn from Coder App warning messages and use them to improve ‘m-file’ coding quality, e.g. complex inputs for SVD(X)
- Carefully consider need to manage occurrences of NaN, Complex, and Infinite input conditions
- Consider creating C code instead of C++ to improve downstream translation efficiency
- Don’t try to rewrite Matlab-only commands so that they ‘act’ like C
- Invest your time in reviewing your Matlab code to understand the impact of transitioning from an interpretive environment to a very low level imperative compiled language
- Study the Coder App options and understand how they map to compiler and linker options
CONCLUSIONS AND RECOMMENDATIONS

▪ If Matlab’s *mobiledev* was extended to incorporate Beacon Eddystone packets, then advanced algorithms on desktop computers running Matlab could access data environments physical contexts worldwide

▪ If Estimote Beacons could be used dynamically, Matlab users would be able to watch their algorithms interacting with the real world

▪ Need Matlab API for Xcode so that iOS apps on iPhone can run Matlab algorithms (via user’s Wi-Fi or data network link)

▪ Matlab needs to rapidly position themselves into the next domain for applications, which is the physical context

▪ Matlab needs to adapt to a physical context-driven environment as opposed to remaining with the default use model, that is, specialized users working on personal computers independent of their physical contexts