C# Orientation Estimation Application Project Document and User Manual

Sommario

Introduction

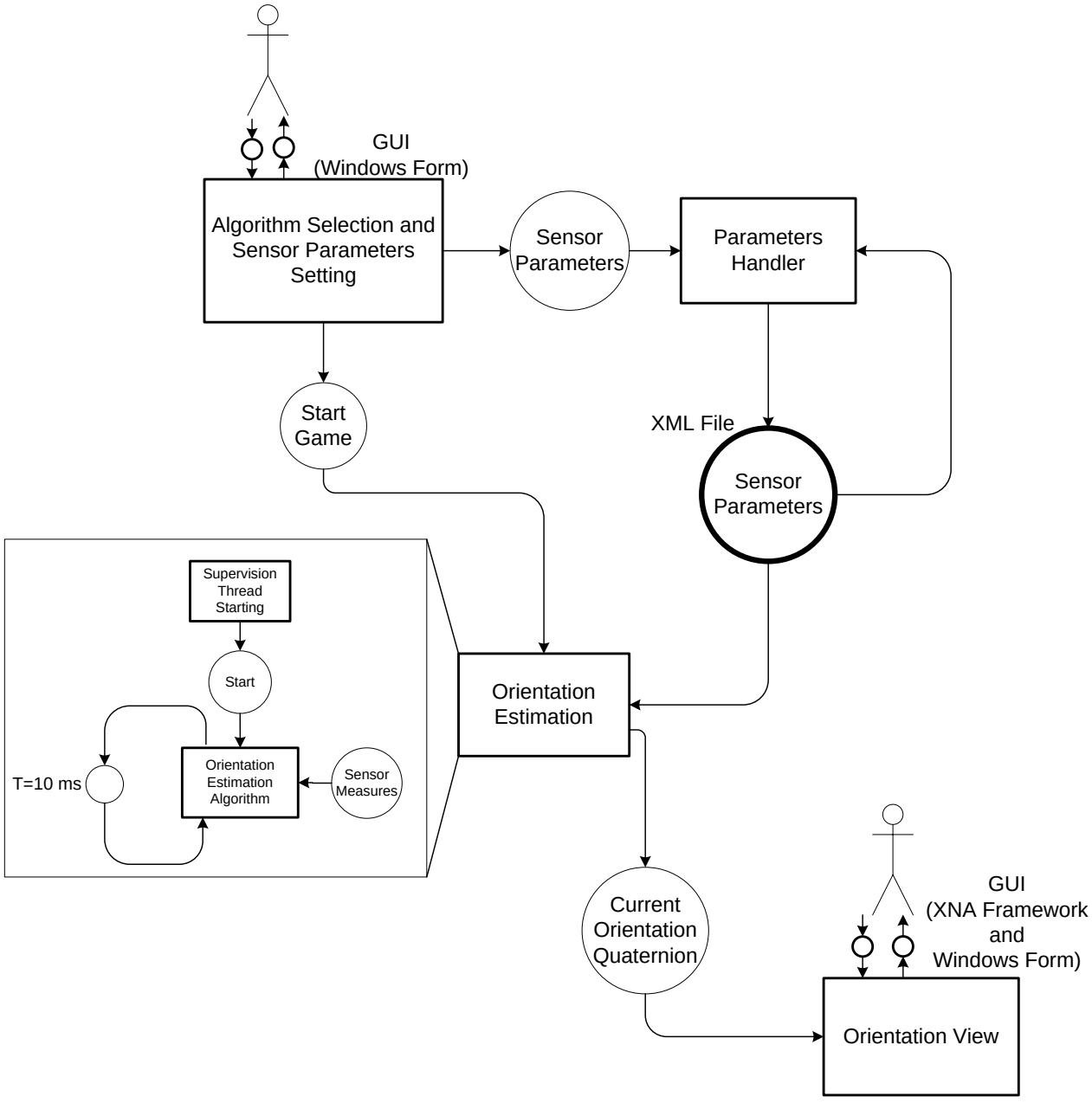
This document contains brief descriptions of the C# software project and a user manual of the orientation estimation application. It is organized as follows.

First chapter briefly treats the software design architecture, while user manual chapter contains useful information about using the software and fit it to the specific IMU.

1 Detailed Design

This project has been created as a “Windows Game” application through Microsoft Visual C# 2008 Express Edition. In order to have a nice user interface, XNA game studio framework has been used.

System dynamics is shown in the following Petri’s network.

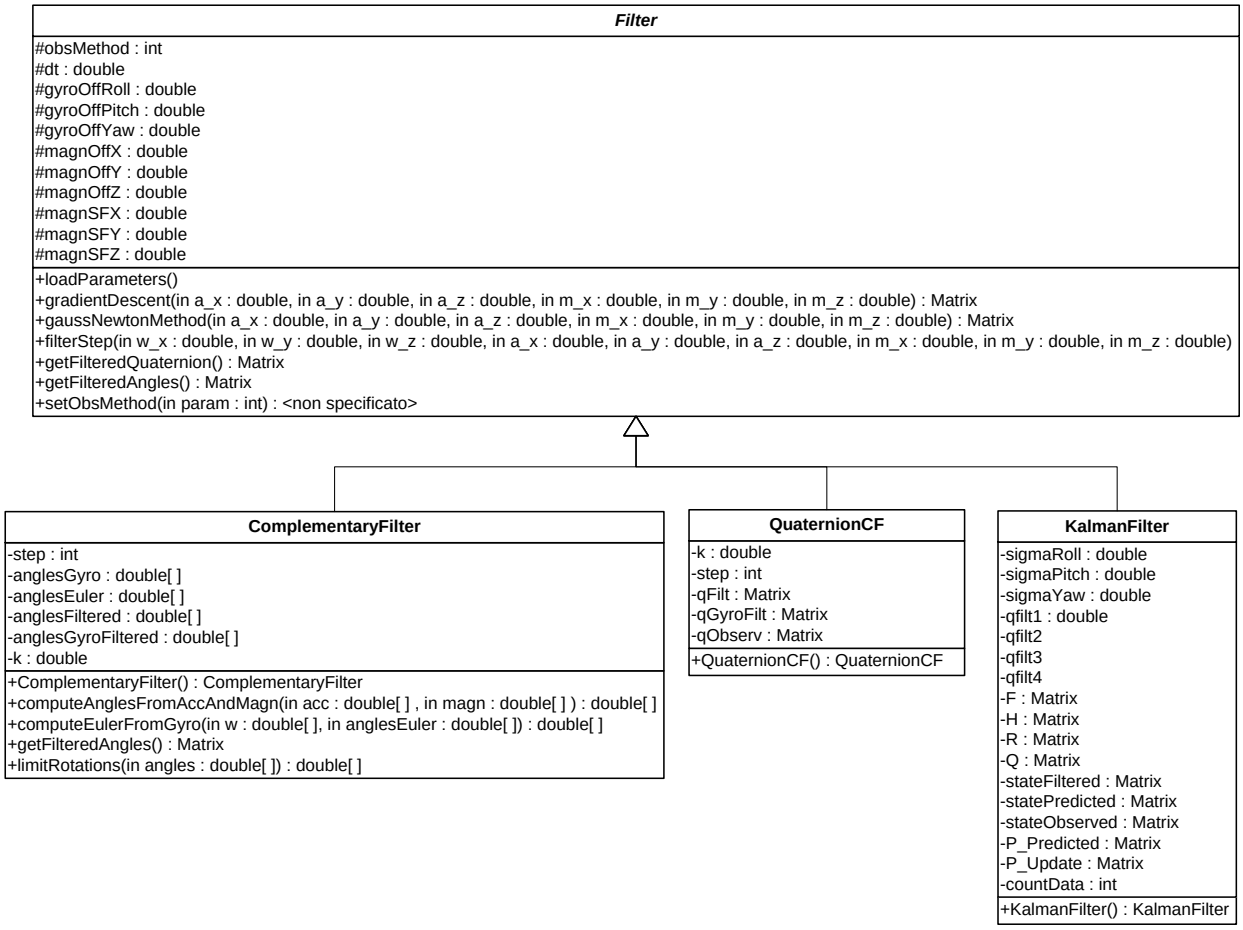


As can be seen, orientation estimation process is timed and handled by a thread, called "AcquisitionThread". This class acquires measurements from the IMU through the C# interface class provided by ST Microelectronics (INEMO2_SDK_Wrapper.cs), which allowed us to avoid a detailed and specific implementation to get measurements.

The GUI components which allows the algorithm selection and the setting up of sensors parameters are subtype of "Windows:Form". These classes are the following:

- SetUpForm.cs, which is the user interface that manage the algorithms selection.
- Parameters.cs, which is the user interface that allows the user to set his specific sensors parameters.

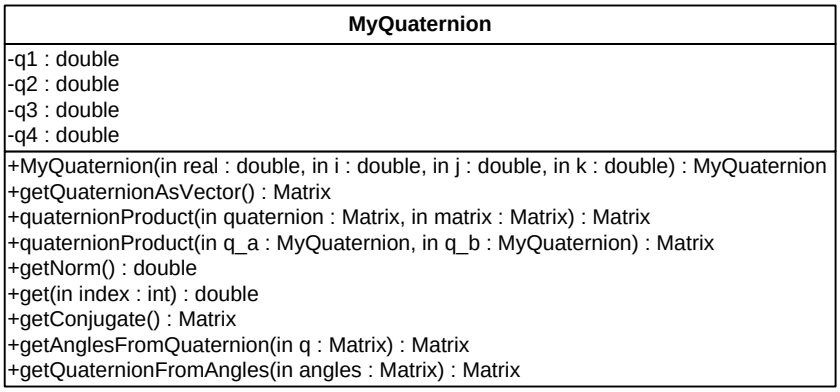
Orientation estimation algorithms architecture is based on a superclass, called "Filter", which implements the main features of a filtering algorithm and which is extended by the specific algorithms, as shown in the figure below.



In the class diagram above basic implemented features, like values of the IIR filters and other properties, are not shown; they can be observed in the C# code. The instantiation of the specific class will take place when the user starts the estimation process.

As can be seen, usage of a class "Matrix" often occurs. This class is achieved by the "cMatrixLib.cs" source code, a library providing support to common matrix operations.

There is also a class implementing quaternion, that fits up to application context requirements. Its static architecture is shown below, in form of class diagram.



In order to allow to any user to change sensors parameters, such as gyroscope and magnetometer offsets and magnetometer scale factors, a persistent resource has been implemented in form of XML file. Its DTD, available in the "debug" directory with the same XML file, is the following:

```
<?xml encoding="UTF-8"?>

<!ELEMENT parameters (magnetometer,gyroscope)>

<!ELEMENT magnetometer (offset,scalefactor)>

<!ELEMENT gyroscope (offset)>

<!ELEMENT offset (#PCDATA)>

<!ELEMENT scalefactor (#PCDATA)>

<!ATTLIST offset axis CDATA #REQUIRED>

<!ATTLIST scalefactor axis CDATA #REQUIRED>
```

In this way, users having more confidence with XML files can directly edit "parameters.xml" file, avoiding to do it through the user interface.

As can be seen in the first "form" opened during application execution, it is possible to plot quaternion trend if a quaternion based algorithm has been selected. This functionality has been achieved using "ZedGraph" libraries, available at the following URL: http://sourceforge.net/projects/zedgraph/.

2 User Manual

2.1 System Requirements

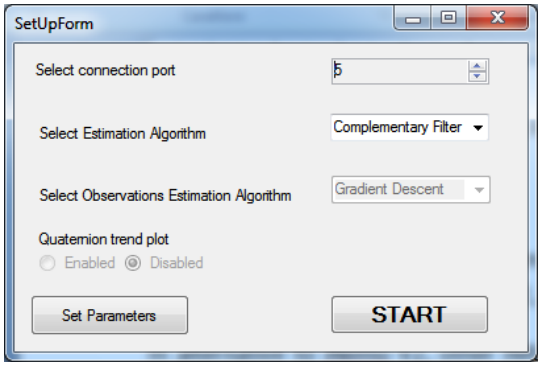
This application is written using XNA Framework and .NET framework. Therefore system requirements are the following:

- OS MS Windows XP or upper versions;
- MS XNA Game Studio Platform Tools / MS XNA Framework 3.1;
- STEVAL-MKI062V2, iNemo V2, as IMU.

In alternative to iNemo V2, other IMU can be used, since these devices are based on the same principles. However, if IMUs different from iNemo are used, it is suggested to edit source code in order to fit it to the IMU specifications.

2.2 Application Usage

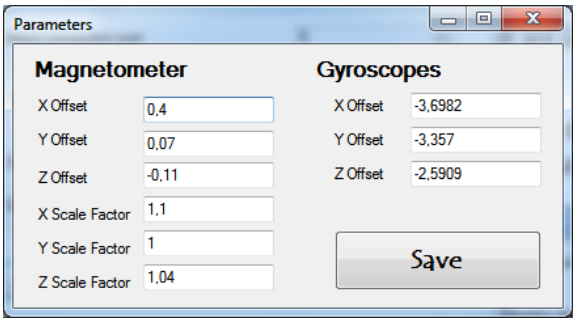
To start application just double click on the "bin\x86\Debug\AngleEstimationApp_BetaRelease.exe". If everything goes well, the following window should appear.



Here the user can choose the following options:

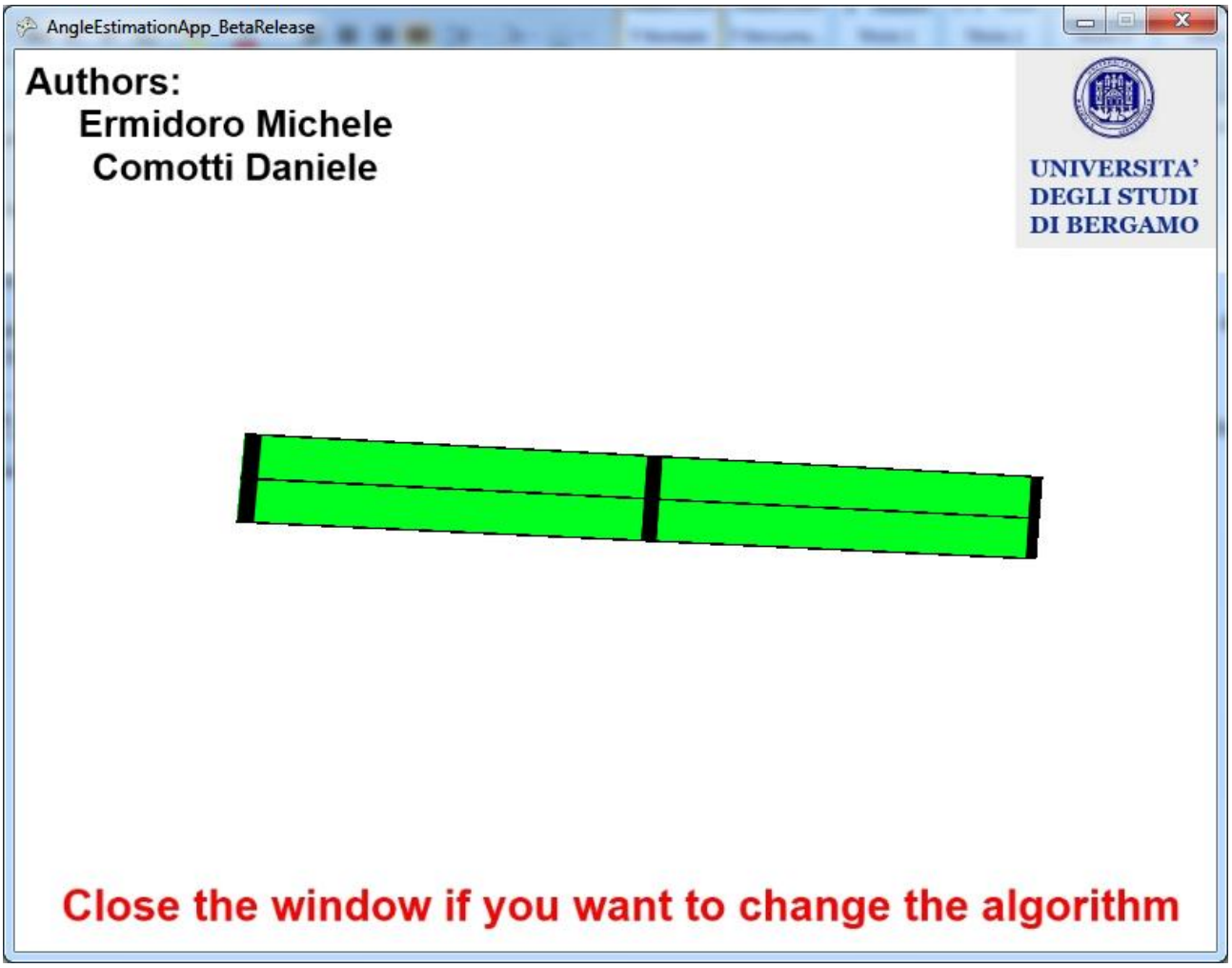
- The connection port, where the IMU is connected;
- The algorithm to use. The following algorithms are available:
    - Complementary Filter, it provides an estimation limited to ± 90 degrees on each angle. Please note that "zero" position is when IMU is headed north.
    - Quaternion Complementary Filter, this algorithm is quite like the previous one, but it can estimate the whole orientation in space.
    - Quaternion Kalman Filter, an alternative algorithm to Quaternion Complementary Filter to get the estimated orientation in space.
- The Observation Estimation method used in the algorithms based on quaternions (this functionality is not available for Complementary Filter);
- The quaternion plot enabling, in order to have a real time trend of the quaternions related to the estimated orientation (this functionality is not available for Complementary Filter).

This application provide a sensors parameters editing, in order to fit the application to the specific iNemo V2 (or IMU) being used. To do it, just click on the "Set Parameters" button. The following window will appear.



Here gyroscope and magnetometer offsets and magnetometer scale factors can be changed. Once the all values has been edited, just click on the "Save" button and exit the window.

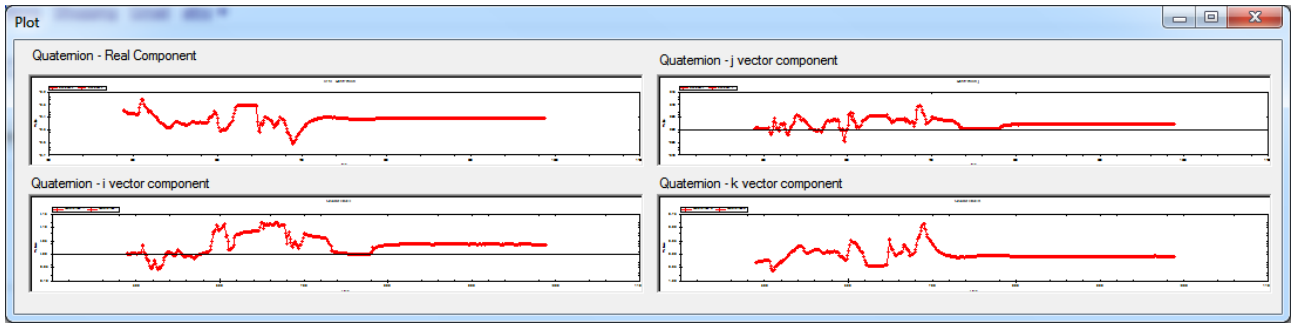
Once all the choices has been taken, just click the "START" button to start the "game". The following window, showing the 3D shape oriented in space and moving like the connected IMU, will appear.



It could happens that the shown shape is not aligned as the IMU. If so, just align the IMU as desired and left click on the mouse. This will align the shape in the same way of the IMU.

To view in full screen, just right click on the window.

If plot of quaternions trend is enabled, the following window will appear.



This window is independent from the one containing the 3D shape, so you can move it as desired.

To stop the “game” just close one of the two windows shown previously.