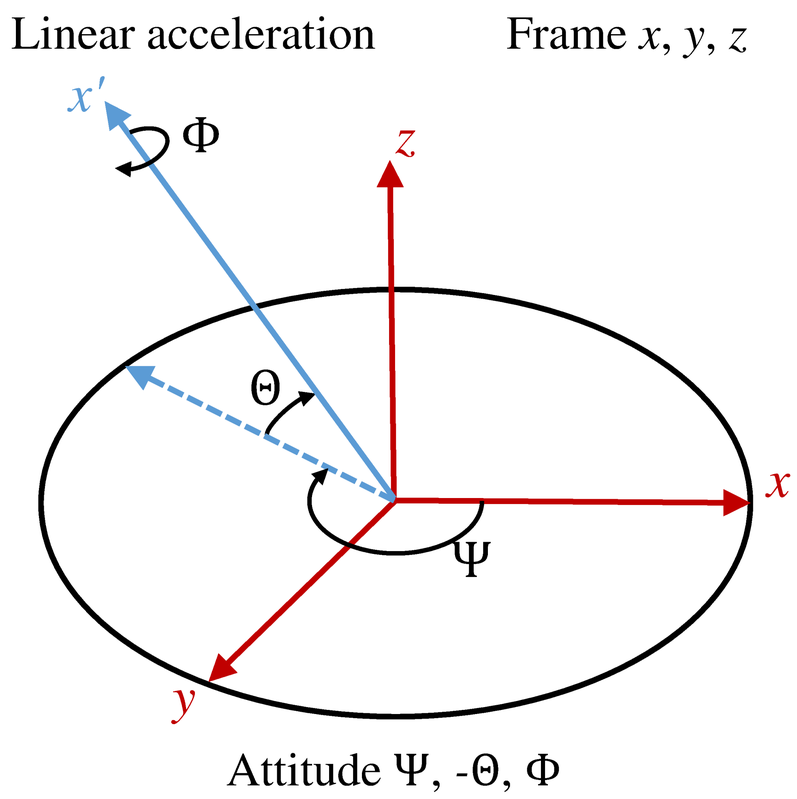
Tampere University

The Direction Cosine Matrix Algorithm in Fixed-Point: Implementation and Analysis

Alexandre Meirhaeghe, Jani Boutellier and Jussi Collin
Faculty of Information Technology and Communication Sciences
***Tampere University**, Finland*

Abstract
Inertial navigation allows tracking and updating the position and orientation of a moving object based on accelerometer and gyroscope data without external positioning aid, such as GPS. Therefore, inertial navigation is an essential technique for, e.g., indoor positioning. As inertial navigation is based on integration of acceleration vector components, computation errors accumulate and make the position and orientation estimate drift. Even though maximum computation precision is desired, also efficiency needs consideration in the age of Internet-of-Things, to enable deployment of inertial navigation based applications to the smallest devices. This work formulates the Direction Cosine Matrix update algorithm, a central component for inertial navigation, in fixed-point and analyzes its precision and computation load compared to a regular floating-point implementation. The results show that the fixed-point version maintains very high precision, while requiring no floating point hardware for operation. The paper presents execution time results on three very different embedded processors.



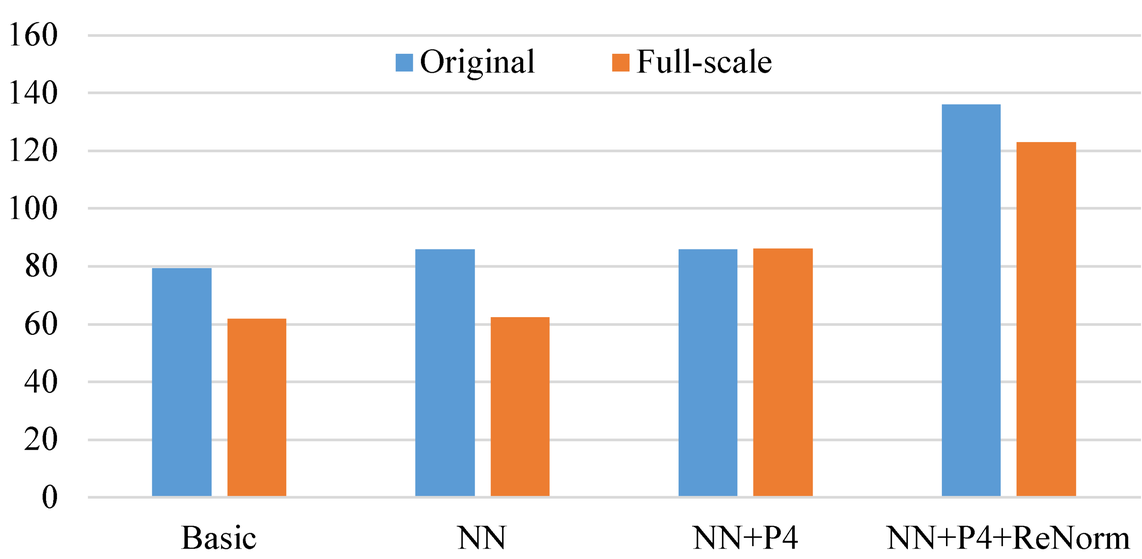
Representation of the coordinate frame, the linear acceleration and the attitude.

Algorithm 1 Direction Cosine Matrix algorithm

1: $\mathbf{in} = \mathbf{gyrodata}\,(i,:)' * \mathbf{T}$
2: $\mathbf{C}_1 = \begin{pmatrix} \mathbf{0} & -\mathbf{in}_3 & \mathbf{in}_2 \\ \mathbf{in}_3 & \mathbf{0} & -\mathbf{in}_1 \\ -\mathbf{in}_2 & \mathbf{in}_1 & \mathbf{0} \end{pmatrix}$
3: $\mathbf{C}_2 = \mathbf{C}_1 * \mathbf{C}_1$
4: $\mathbf{p} = \sqrt{\mathbf{in}' * \mathbf{in}}$
5: $\mathbf{out} = \mathbf{I} + \frac{\sin(\mathbf{p})}{\mathbf{p}} * \mathbf{C}_1 + \frac{1-\cos(\mathbf{p})}{\mathbf{p}^2} * \mathbf{C}_2$
6: $\mathbf{DCM} = \mathbf{DCM} * \mathbf{out}$

Algorithm 2 Direction Cosine Matrix algorithm in fixed point

1: $\mathbf{in} = \mathbf{gyrodata}\,(i,:)'$ **Q6.26**
2: $\mathbf{C}_1 = \begin{pmatrix} \mathbf{0} & -\mathbf{in}_3 & \mathbf{in}_2 \\ \mathbf{in}_3 & \mathbf{0} & -\mathbf{in}_1 \\ -\mathbf{in}_2 & \mathbf{in}_1 & \mathbf{0} \end{pmatrix}$ **Q6.26**
3: $\mathbf{C}_2 = \begin{pmatrix} \mathbf{C}_{11} * \mathbf{C}_{13} + \mathbf{C}_{12} * \mathbf{C}_{16} & \mathbf{C}_{12} * \mathbf{C}_{17} & \mathbf{C}_{11} * \mathbf{C}_{15} \\ \mathbf{C}_{15} * \mathbf{C}_{16} & \mathbf{C}_{13} * \mathbf{C}_{11} + \mathbf{C}_{15} * \mathbf{C}_{17} & \mathbf{C}_{13} * \mathbf{C}_{12} \\ \mathbf{C}_{17} * \mathbf{C}_{13} & \mathbf{C}_{16} * \mathbf{C}_{11} & \mathbf{C}_{16} * \mathbf{C}_{12} + \mathbf{C}_{17} * \mathbf{C}_{15} \end{pmatrix}$ **Q10.22**
4: $\mathbf{p}_2 = \mathbf{in}_3 * \mathbf{in}_3 + (\mathbf{in}_2 * \mathbf{in}_2 + \mathbf{in}_1 * \mathbf{in}_1) >> \mathbf{1}$ **Q10.22**
5: $\mathbf{p}_4 = \mathbf{p}_2 * \mathbf{p}_2$ **Q19.13**
6: $\mathbf{Cff}_1 = \mathbf{2}^{31} + \left(\left(\mathbf{p}_4 * \frac{\mathbf{2}^{64}}{\mathbf{120}}\right) >> \mathbf{9} - \mathbf{p}_2 * \frac{\mathbf{2}^{46}}{\mathbf{6}}\right) >> \mathbf{7}$ **Q1.31**
7: $\mathbf{Cff}_2 = \mathbf{2}^{31} + \left(\left(\mathbf{p}_4 * \frac{\mathbf{2}^{67}}{\mathbf{720}}\right) >> \mathbf{9} - \mathbf{p}_2 * \frac{\mathbf{2}^{48}}{\mathbf{24}}\right) >> \mathbf{8}$ **Q0.32**
8: $\mathbf{C100} = \frac{\mathbf{2}^{37}}{\mathbf{100}}$ **Q − 5.37**
9: $\mathbf{C_2Cff_2} = \mathbf{C}_2 * \mathbf{Cff}_2 * \mathbf{C100}$ **Q3.29**
10: $\mathbf{out} = \mathbf{C}_1 * \mathbf{Cff}_1 + (\mathbf{C_2Cff_2}) >> \mathbf{3}$ **Q6.26**
11: $\mathbf{out} = \mathbf{I} + (\mathbf{out} * \mathbf{C100}) >> \mathbf{2}$ **Q1.31**
12: $\mathbf{DCM} = \mathbf{DCM} * \mathbf{out}$ **Q1.31**
13: $\mathbf{DCM} = \mathbf{DCM} + \frac{(\mathbf{I} - \mathbf{DCM} * \mathbf{DCM}') * \mathbf{DCM}}{\mathbf{2}}$ **Q1.31**



Accuracy of the results as signal-to-noise ratio (in dB scale).

Device	*Achievable sample rate*
Microchip ATmega 2560 (8-bit)	0.275 kilosamples/s
SiFive Freedom E310 (32-bit)	12.68 kilosamples/s
Qualcomm Hexagon 682	2148 kilosamples/s

Processing time results of *NN+P4+Renorm* on various platforms.