# Improving feature tracking using motion sensors

Jean-Baptiste Boin
Stanford University
jbboin@stanford.edu

## 1. Introduction

Porting state-of-the-art computer vision algorithms to mobile devices is a difficult, but desirable task. Indeed, some applications require mobility, like augmented reality. However, running an application in real time may be tricky given the limited resources available on a mobile device. This is why the algorithms may need to be downscaled or simplified.

A typical augmented reality pipeline is composed of object recognition, object localization and object tracking. In this project, we will be focusing on the third part, the tracking of a target in real-time. We will be focusing on feature-based tracking, and our goal will be to make it more efficient by using the motion sensors in the mobile device (gyroscopes, accelerometers, magnetometer). Our approach will be compared to a pure vision-based feature matching scenario, which will be our baseline. Since we want to be able to track our target in real-time, we need to run the tracking algorithm for each frame and it should complete before the next frame is acquired. This means that achieving an efficient tracking could be critical to the performance of an augmented reality application. If the baseline can be efficient enough for simple situations, it is possible that some optimizations are needed if we need to deal with more complex objects (e.g. several 2D targets or one 3D target). In that case, an increase in the performance would allow for more advanced applications. For this project, we will try to develop and evaluate such an improvement in simple situations.

## 2. Technical details

### 2.1. Sensors

As presented in the Google Tech Talk "Sensor Fusion on Android Devices: A Revolution in Motion Processing"[1], the noise in the orientation angles and in the linear acceleration account for a very large drift once these accelerations are double-integrated, and so it makes it completely impossible to measure translation between two camera poses by using just the sensors of the device. However, it seems to be possible to measure rotations with reasonable accuracy, and we have several tools available for this purpose.

The Android SDK allows us to access values from different sensors in the device, but it also offers interesting "composite" sensors that are based on values from these sensors.[2] The ones that could be interesting for us are the following:

- *Rotation vector* (uses accelerometer, gyroscope and magnetometer) : gives the rotation of the device with respect to a fixed coordinate system, East-North-Up.
- *Game rotation vector* (uses accelerometer and gyroscope) : same as the previous one except that the direction of the North is unknown so the X axis points in an unknown reference direction. This reference direction can drift since the gyroscope can drift around the Z (vertical) axis.
- *Geomagnetic rotation vector* (uses accelerometer and magnetometer) : same as the rotation vector composite sensor except that it does not use the gyroscope to refine its estimation.

These composite sensors are obtained from the values given by the raw sensors and processing them using Kalmann filters, commonly used in sensor fusion.

We already eliminate the third choice because the accelerometer and magnetometer only give reliable estimations for low frequency motion. The gyroscope gives good estimates for high frequency motion, so it makes sense to use it if we want to be able to track motion in real time. It is not clear if we prefer to use the first or the second composite sensor, and additional experiments will be necessary to know if the magnetometer is reliable enough to be used for our application. If it is, the rotation vector sensor will be a sensible choice, but otherwise the game rotation vector sensor will be more appropriate, although the drift may cause additional problems we will have to deal with.

### 2.2. Motion field

We will now go through the theory of motion field to see how the knowledge of the rotation between two frames can help us with the tracking of features.

We are given a fixed 3D point $\mathbf{X}$ and a camera with camera matrix $[\mathbf{R}, \boldsymbol{t_c}]$ and focal length $f$. The point projects to $\boldsymbol{x} = \frac{f\boldsymbol{X}}{Z}$ on the camera image plane. We assume that we know the linear velocity of the camera, $\mathbf{T}$, and its angular velocity, $\boldsymbol{\Omega}$.

Now, we want to estimate the velocity $\mathbf{v}$ of $\mathbf{x}$ given the linear and angular velocities of the camera.

$$\boldsymbol{v} = \dot{\boldsymbol{x}} = f\frac{Z\boldsymbol{V} - V_z\boldsymbol{X}}{Z^2}$$

where $\mathbf{V}$ is the velocity of the 3D point with respect to the camera.

We can estimate this velocity using the infinitesimal motion of the camera between a time $t$ and $t + dt$

$$\begin{aligned}\boldsymbol{V}dt &= \boldsymbol{X} - [d\boldsymbol{R}, \boldsymbol{T}dt]\begin{pmatrix}\boldsymbol{X}\\1\end{pmatrix}\\ &= -\boldsymbol{T}dt + (\boldsymbol{I} - d\boldsymbol{R})\boldsymbol{X}\end{aligned}$$

We can relate $d\boldsymbol{R}$ to $\boldsymbol{\Omega}$ by using the following equation:

$$d\boldsymbol{R} = \boldsymbol{I} + [\boldsymbol{\Omega}dt]_\times$$

This yields, after division by $dt$

$$\boldsymbol{V} = -\boldsymbol{T} - \boldsymbol{\Omega} \times \boldsymbol{X}$$

Finally, by substituting this expression for $\boldsymbol{V}$ in our first equation, we get the velocitied of the image point

$$\begin{aligned}v_x &= \frac{T_z x - T_x f}{Z} - \Omega_y f + \Omega_z y + \frac{\Omega_x xy - \Omega_y x^2}{f}\\ v_y &= \frac{T_z y - T_y f}{Z} + \Omega_x f - \Omega_z x + \frac{\Omega_x y^2 - \Omega_y xy}{f}\end{aligned}$$

This expression is quite interesting because the left part only uses translation information (as well as the depth of the point), while the right part only requires rotational information. If we do not know a priori the depth of the object, nor the translation of the camera (cf. previous paragraph), then we can treat this left part as noise, while the right part can be obtained from the rotation of the device. This theoretical framework justifies this whole project since we can see that some knowledge on the motion of the camera directly translate in the evolution of the position of a feature over time.

### 2.3. Ambitions for the project

We will be working on the Tegra Note tablet borrowed for this class.

The first part of the project will be to experiment with the different components we will use during the project. This will allow us to see how stable the "rotation vector" composite sensor is when we move the camera around, and this insight will definitely be useful in the choices we will make in the subsequent steps.

The next step will be to integrate the sensor measurements with OpenCV. In this step, we will be able to actually estimate how useful is the knowledge of the rotation between two frames. We will proceed this way. First we extract FAST or ORB features from a frame. These features can be extracted at frame rate which is necessary for our matching. Then, in a next frame (it could be the one just after, or N frames later), we also extract the same features, that we match using the standard OpenCV matching functions. We also get rid of the outliers using geometric verification. Given the relative rotation, we can compute where the features of the first frame should be located in the second frame. This will give us an idea of the errors that we would get from using the rotation given the sensors. We will run our experiments with pure rotations, by trying to rotate the device around the camera and by choosing targets that are sufficiently far away. Experiments with translations or close objects could also give interesting information on how the errors vary if we have time for that.

As a third step, we will try to create the fast tracker that is our final desired outcome. There are two ways that we can use to make the tracking more efficient.

- *Geometric verification:* If we know how the features should have moved between two frames, then we can use this knowledge to get rid of the outliers after matching, if the motion is too different to the expected one. This would perhaps allow us to get rid of the whole geometric verification RANSAC loop.

- *Matching*: If we know the position of a tracked feature in a new frame, as well as the accuracy of this estimate, we can decrease the time taken for the feature matching by just matching the tracked feature with the new features found in a neighborhood or this position.

We will consider the first optimization, which sounds like the one that will give us the best improvement in terms of time. If time allows it, we will improve the second optimization. However, it is not obvious that this improved matching will give us a noticeable difference compared to the full matching. In the case where translations should also be considered, the uncertainty zone to match each feature with will be larger and so there may not even be any improvement in matching a tracked feature with only a restricted number of features. This is a question we will try to answer.

A fourth step could be added if we want to push this project further: in an augmented reality framework, we can track an actual target, for example a 2D target printed on paper, and deal with the questions of drift in the tracking: would it be better to track the target by using direct

matching (matching with a reference one) or by incremental matching (matching with the previous frame)?

## 3. Milestones

We do not have much time for this project so we try to keep our ambitions reasonable. In the three weeks remaining before the deadline, we will first spend the first ten days to complete step one and two (getting familiar with the sensors and comparing the theoretical results to the actual ones). The remaining ten days will be used for going as far as possible on step three, our priority being to see if we can safely remove the geometric verification step.

## 4. Thanks

I would like to thank Roland Angst for his help on the theoretical background, as well as for committing to be an advisor for this project.

## References

[1] D. Sachs, "Sensor Fusion on Android Devices: A Revolution in Motion Processing", 2010, accessible at http://youtu.be/C7JQ7Rpwn2k.

[2] Android SDK documentation, "Composite sensors" page, accessible at https://source.android.com/devices/sensors/composite_sensors.html.