Efficient Online Egomotion Estimation using Visual and Inertial Readings

Raphael Mannadiar
Center for Intelligent Machines, McGill University
raphael.mannadiar@mail.mcgill.ca

Abstract

An egomotion estimator that makes use of the complementary strengths of inertial and visual readings is introduced in the context of efficient and robust real-time motion estimation. Our work is targeted to a noisy, computationally limited and unconstrained environment. Experimental results show that the proposed technique, despite being built upon heuristics that often privilege speed over exhaustiveness, produces extremely precise and robust egomotion estimates in situations that cripple visual-only and inertial-only trackers.

1 Introduction

Egomotion estimation consists in determining one self's motion. Humans do this remarkably well by fusing several sources of information e.g. their eyes, their sense of balance, their knowledge of the world and so on. It can be argued and past research has established that combining visual and inertial sensor readings to determine egomotion yields much better results than those obtained using only one type of reading. This result is both intuitive and has been empirically demonstrated repeatedly [16] [17] [15] [1]. It seems that cameras and inertial measurement units (IMU) complement each other very well : for discontinuous motions where the visual feature tracker fails to make sense of incoming images, the IMU excels, whereas in slower smoother motions where the IMU's readings are drowned in noise, the visual feature tracker performs well.

The synergy obtained by combining these two types of measurement devices has prompted much research on the topic. Section 2 provides a brief overview of the commonalities and differences of published approaches. Section 3 presents a new approach that combines new ideas with those from the literature with the objective of producing an estimation algorithm that could run in real-time on properly equipped portable consumer electronic devices (PCED), such as *Apple's iPhone Touch* for instance. Hence, our technique aims not only to produce correct results in a prompt and robust manner, it must also take into account the poor quality of sensors that could viably (in terms of cost and size) be packaged with PCEDs, the computational limitations of such devices and the unbounded variety of sceneries that they can be exposed to. Section 4 presents a thorough evaluation of the approach based on simulated data. Finally, Sections 5 and 6 cover future work and conclusions.

2 Related Work

Several authors make use of statistical filters to fuse inertial and visual readings [16] [17] [15] [1], many of which produce noteworthy results. For instance, [15] and [1] use Kalman filters (or Kalman filter extensions) to integrate measurements produced by both their measurement devices whereas [16] and [17] use Kalman filters to estimate egomotion, inertial readings, estimation errors and the 3D coordinates of all known features. The latter approach seems somewhat out of place in the context of this work, i.e. in a real-time context targeting devices with limited computing power or battery life that might suffer from having to crunch through "large" matrix computations. Furthermore, though different approaches may have different constraints and motivations, in this work we privilege those that give more care to analyzing the specificities of the problem at hand than is possible with the usage of a generic number cruncher.

Other research attempts to vary the nature of visual readings using edges [8] [13], countours [1] or optical flow rather than points. Though the use of countours may seem appealing, current algorithms for contour

detection are often clumsy, costly and complex especially when one wants to build in robustness against occlusions and background clutter and that no prior knowledge about the observed scenes is provided [2] [20].

Another technique that has had some success is the biasing of the world via model-based visual tracking (MBVT) [8] [13] [17]. Though there may be many situations where optimizing an algorithm for usage in some specific environment (e.g. office spaces, warehouses, etc.) is a viable or desirable solution, this is clearly not the case in the context where the operator is free to roam the world. The results of papers that make use of MBVT, though often very impressive, lack meaning in the context of producing a commercial product that could keep track of its own motion in virtually any environment and situation.

3 Approach

Our approach operates without a prior world model, and attempts to avoid costly statistical filtering. Like almost all methods, we do employ an iterative estimator; i.e., as sensor readings come in, we

1. Estimate the expected motion;
2. Measure the actual robot motion;
3. Update the current pose estimate;
4. Update feature point coordinates.

Our key contributions are the introduction and combination of a set of heuristics that allow (1) the efficient and robust tracking of visual features and egomotion estimation even in erratic situations, and (2) the filtering of noise from inertial readings to produce an egomotion system that is virtually unaffected by drift. These heuristics will be covered in detail in the following subsections.

3.1 Estimating expected motion

We determine the estimated egomotion since the last reading by computing motion metrics such as linear and angular velocities [1] from the the accumulated sequence of poses – the initial pose is centered at the origin looking straight down the Z axis – and applying Newton's laws of motion :

$$\vec{d}_{t+\Delta t} = \vec{v}_t * \Delta t$$
$$\vec{\phi}_{t+\Delta t} = \vec{\omega}_t * \Delta t$$

where $\vec{d}_{t+\Delta t}$ and $\vec{\phi}_{t+\Delta t}$ are respectively the predicted 3D linear and angular egomotions for time $t + \Delta t$, and $\vec{v}_t$ and $\vec{\omega}_t$ are respectively the 3D linear and angular velocities from the pose at time $t - \Delta t$ to the one at time t. Empirical tests demonstrate that motion predictions produced in this manner are typically erratic due to the influence of noise in the readings. To reduce the motion prediction volatility, a simple smoothing function is applied that integrates predicted motions for times t and $t-\Delta t$ with the predicted motion for time $t+\Delta t$ yielding $\vec{\bar{d}}_{t+\Delta t}$ and $\vec{\bar{\phi}}_{t+\Delta t}$:

$$\vec{\bar{d}}_{t+\Delta t} = \alpha\vec{d}_{t+\Delta t} + \beta\vec{\bar{d}}_t + (1-\alpha-\beta)\vec{\bar{d}}_{t-\Delta t}$$

$$\vec{\bar{\phi}}_{t+\Delta t} = \alpha\vec{\phi}_{t+\Delta t} + \beta\vec{\bar{\phi}}_t + (1-\alpha-\beta)\vec{\bar{\phi}}_{t-\Delta t}$$

where $\alpha > \beta$ and $\alpha + \beta + (1-\alpha-\beta) = 1$. On a final note, since the motion of the features is opposite to that of the camera – e.g. if the camera moves to the right, the features will move to the left –, the predicted camera and feature motions respectively become [2] :

$$< \tilde{M}^c_{t+\Delta t}.\vec{T}, \tilde{M}^c_{t+\Delta t}.\vec{R} > = < \vec{\bar{d}}_{t+\Delta t}, \vec{\bar{\phi}}_{t+\Delta t} >$$

$$< \tilde{M}^f_{t+\Delta t}.\vec{T}, \tilde{M}^f_{t+\Delta t}.\vec{R} > = < -\vec{\bar{d}}_{t+\Delta t}, -\vec{\bar{\phi}}_{t+\Delta t} >$$

3.2 Measuring actual robot motion

The computation of egomotion from incoming readings is quite different depending on the nature of the readings – i.e. inertial versus visual; both scenarios are developed below.

3.2.1 Inertial readings

An inertial reading is a 6-degree vector of the form $< \vec{O}, \vec{A} >$ where $\vec{O}$ and $\vec{A}$ respectively denote 3D inertial sensor orientation [3] and linear acceleration. We compute meaningful motion information in two steps: extraction and filtering. Let I be an incoming inertial reading at time t.

[1] Inter-reading arrival time is assumed sufficiently short for linear and angular accelerations to be ignored

[2] Throughout this work, the notation X.Y denotes the attribute Y of entity X

[3] This work assumes that the camera and inertial sensor share the same axes of alignment i.e. that camera and sensor orientation are identical.

First, the "raw" motion information is extracted from the reading. Newton's equations of motion are used to obtain the linear motion whereas as a simple difference is used to estimate the angular motion:

$$M_t^{ri}.\vec{T} \;=\; \vec{v}_t\Delta t \;+\; \frac{\Delta t^2}{2} I.\vec{A}$$

$$M_t^{ri}.\vec{R} \;=\; I.\vec{O} \;-\; \vec{O}_{t-\Delta t}$$

where $\vec{O}_{t-\Delta t}$ is the sensor – or pose – orientation at time $t-\Delta t$, $I.\vec{O}$ and $I.\vec{A}$ are respectively the orientation and linear acceleration components of I, and M_t^{ri} is the "raw" motion information described by I.

Second, to reduce the effect on measurement noise, we use a Kalman filter-based approach. The idea is to use the variance-based merging techniques from the Kalman Update step to merge M_t^{ri} with $\tilde{M}_t^c$ to produce an optimal estimate of the actual motion that occurred, $\breve{M}_t^i$:

1. *Setup prediction and measurement variances*: The prediction variance ρ has a fixed and fairly low value that encodes the fact that the accumulated sequence of poses is always deemed trustworthy. As for the measurement variance $M_t^{ri}.\vec{\sigma}$, it is a 6-degree vector that relates the magnitude of the reading components to the sensor accuracy such that as reading component magnitudes increase, their variances decrease.

2. *Go through the Kalman Update step*: This implies computing the Kalman Gain and updating the motion estimate. In Kalman filter terms, using the variances defined in step 1 as the estimated state error and measurement covariances and $\tilde{M}_t^c$ and M_t^{ri} as the state estimate and measurement respectively, we can compute the Kalman Gain and produce an updated state estimate. See below for the Kalman Gain and updated state estimate equations.

$$\vec{K}_{t_i} \;=\; \frac{\rho}{\rho + M_t^{ri}.\vec{\sigma}_i}$$

$$\breve{M}_t^i.\vec{R} \;=\; \begin{pmatrix} \tilde{M}_t^c.\vec{R}_x + K_{t_1} * (M_t^{ri}.\vec{R}_x - \tilde{M}_t^c.\vec{R}_x) \\ \tilde{M}_t^c.\vec{R}_y + K_{t_2} * (M_t^{ri}.\vec{R}_y - \tilde{M}_t^c.\vec{R}_y) \\ \tilde{M}_t^c.\vec{R}_z + K_{t_3} * (M_t^{ri}.\vec{R}_z - \tilde{M}_t^c.\vec{R}_z) \end{pmatrix}^T$$

$$\breve{M}_t^i.\vec{T} \;=\; \begin{pmatrix} \tilde{M}_t^c.\vec{T}_x + K_{t_4} * (M_t^{ri}.\vec{T}_x - \tilde{M}_t^c.\vec{T}_x) \\ \tilde{M}_t^c.\vec{T}_y + K_{t_5} * (M_t^{ri}.\vec{T}_y - \tilde{M}_t^c.\vec{T}_y) \\ \tilde{M}_t^c.\vec{T}_z + K_{t_6} * (M_t^{ri}.\vec{T}_z - \tilde{M}_t^c.\vec{T}_z) \end{pmatrix}^T$$

The measurement variance definitions take into account the fact that IMU readings are more meaningful in erratic situations than in smaller smoother motions. An example of this is that of a drastic directional change: some or all components of the inertial reading will considerably exceed the device accuracy producing low variances for these components which in turn will lead to $\breve{M}_t^i$ being more similar to M_t^{ri} than to $\tilde{M}_t^c$. A very different example is that of a stationary system where $\tilde{M}_t^c$ should be near null whereas M_t^{ri} might describe some small measured motion due to noise in the inertial reading: here, the components of the reading should have a magnitude similar to the device accuracy producing high variances which in turn will lead to $\breve{M}_t^i$ being more similar to $\tilde{M}_t^c$ than to M_t^{ri}. Finally, before computing $\breve{M}_t^i$ as the measured motion at time t, we associate a variance to it that we will use later when updating the pose estimate. This variance is selected as the smallest value in $M_t^{ri}.\vec{\sigma}$ or as a "very high" number – within bounds]0,1[[4] – if all values in $M_t^{ri}.\vec{\sigma}$ exceed 1. This is yet another mechanism that ensures only meaningful inertial readings are allowed to contribute to the pose estimate and that their contribution is measured by their degree of certainty.

3.2.2 Visual readings

A visual reading is a list of feature points in image space, i.e. an N-element vector of 2D coordinates of the form $< (u_1, v_1),\ (u_2, v_2),\ ...,\ (u_N, v_N) >$. Again, we compute meaningful motion information in two steps: tracking and solving. Let V be an incoming visual reading that arrives at time t.

First, we establish correspondences between known features and the "new" features in V, i.e. old features $F(X, Y, Z)$ are assigned to corresponding new features $f(u, v)$. We use two complementary tracking algorithms to accomplish this task. The first one, which will be referred to as the *UV-Tracker*, is very efficient but its robustness is somewhat dependent on the accuracy of the predicted motion. On the other hand, the second tracking algorithm, which will be referred to as the *Z-Tracker*, is much less efficient but more robust to errors in the predicted motion. Both algorithms are detailed below.

The *UV-Tracker*: The goal of this algorithm is to reduce the computational complexity and image space search ranges of feature tracking. We accomplish this

[4] Variances throughout this work are bounded by 0 and 1

via a heuristic that uses the predicted motion of the features, $\tilde{M}_t^f$, to focus the matching effort on small regions of the image space. The main steps of the algorithm follow.

1. Sort new features into range tree T;
2. For each old feature F,
2(a). Apply $\tilde{M}_t^f$ to obtain $\tilde{F}_t$, the predicted position of F at time t;
2(b). Project [5] $\tilde{F}_t$ onto image space to obtain $\tilde{F}'_t$;
2(c). Establish bounds in UV space around $\tilde{F}'_t$ in which corresponding new features will be searched for. To account for the fact that features are more affected by camera motion as they get closer to it, these bounds are inversely proportional to $\tilde{F}_{t_z}$:

$$\tau \;=\; MAX(\,\varepsilon\,,\;\left|\frac{\lambda}{\tilde{F}_{t_z}}\right|)$$

$$\tilde{U} \;=\; \begin{pmatrix}\tilde{F}'_{t_u}-\tau\\ \tilde{F}'_{t_u}+\tau\end{pmatrix}^T$$

$$\tilde{V} \;=\; \begin{pmatrix}\tilde{F}'_{tv}-\tau\\ \tilde{F}'_{tv}+\tau\end{pmatrix}^T$$

where τ is the UV search radius (lower-bounded by ε pixels and with a value of λ pixels for an object 1 meter away) and $\tilde{U}$ and $\tilde{V}$ are respectively the U- and V-search bounds. In our experiments, we use ε=5 and λ=30. As such, ε is sufficiently large (and small) to serve only its purpose: to account for noise in the visual reading that can cause a feature to falsely appear to have moved in a motionless context. As for λ, our tests show that the chosen value offers a good compromise between robustness to prediction errors and tracking efficiency.
2(d). Query T for features within the computed bounds and mark the feature, if any, closest to $\tilde{F}'_t$ in Euclidean distance.
3. To attenuate any ill effect false matches may have on the upcoming solving step, any matches involving new features that were matched to more than one old feature are discarded;
4. Finally, if more than 40% of new features were matched, move on to the solving step; if not, move to the *Z-Tracker*.

The *Z-Tracker*: The *UV-Tracker* algorithm, with its U- and V-search radii, is designed for robustness to errors in the X and Y coordinates of $\tilde{F}_t$. However, noisy predicted motions – especially noisy X and Y angular motion components – sometimes cause $\tilde{F}_{t_z}$ to be inaccurate. In such cases, projecting $\tilde{F}_t$ onto the image space yields points that are too far away from their corresponding new feature to be properly matched. Moreover, this problem often affects the tracking of nearly all of the known features because the same noisy predicted feature motion is applied to all of them. The goal of the *Z-Tracker* algorithm is to overcome noise in $\tilde{M}_t^f$. It does so in a manner somewhat inspired from search norms in countour tracking [2]. Rather than search for matching new features in the U-, V-vicinity of $\tilde{F}'_t$, it searches for new features that would "most easily" project onto F's projection in image space. The detailed steps of the algorithm follow.

1. For each old feature F,
1(a). For each new feature f, determine $\hat{z}$ such that $\hat{f}$ and F are closest in Euclidean distance; where

$$\hat{f} \;=\; \begin{pmatrix}\frac{f_u * \hat{z}}{\gamma} & \frac{f_v * \hat{z}}{\gamma} & \hat{z}\end{pmatrix}$$

where γ is the camera focal length. In short, we are searching for a feature match along the Z-axis rather than along the X- and Y-axes by computing the Z coordinate that would place f closest to F in 3D space without modifying its projection point in image space. Solving

$$MIN_{\hat{z}}\; distance(\;\hat{f}\;,\;F\;)^2 \tag{3.1}$$

yields the optimal $\hat{z}$

$$\hat{z} \;=\; \frac{\left(\frac{2F_x * f_u}{\gamma} + \frac{2F_y * f_v}{\gamma} + 2F_z\right)}{\left(2\frac{f_u^2}{\gamma^2} + 2\frac{f_v^2}{\gamma^2} + 2\right)}$$

2. Let f^* be the new feature that minimizes Eq 3.1 over all f;
3. Same as *UV-Tracker*'s 3rd step.

While this tracking algorithm is somewhat costly, its main purpose is to help the system recover from noisy predicted motions and as such, robustness, not efficiency, is the priority. Nevertheless, future work will explore reducing the search space.

Empirical testing of the overall technique presented in this work reveals that the *UV-Tracker* and the *Z-Tracker* have complementary failing scenarios, a fact which makes the two algorithms very proficient partners despite their respective flaws.

Second, given the set of matches reported by the feature tracking step, we use a minimization algorithm to

[5] Using the perspective projection model

find the most likely motion that would have caused the old feature points to land on their equivalent new selves. This step is necessary because although this most likely motion is often expected to be quite similar to the predicted motion, it may differ. The minimization algorithm used here is a tuned version of the Levenberg-Marquardt (LM) algorithm tailored to solve equations of the form

$$\sum_{i=1}^{N} \left((y_i - f(x_i, \beta))^2 + (z_i - g(x_i, \beta))^2 \right)$$

In this case, the equation to solve over the set of all possible motions is

$$\sum_{i=1}^{N} distance(\dot{F_i}', f_i^*)^2$$

$$= \sum_{i=1}^{N} \left((\dot{F_{i_u}}' - f_{i_u}^*)^2 + (\dot{F_{i_v}}' - f_{i_v}^*)^2 \right) \qquad (3.2)$$

where $< F_i, f_i^* >$ is the i^{th} of N reported feature matches, $\dot{F_i}$ is the point obtained by applying motion M to feature F_i, $\dot{F_i}'$ is the projection of $\dot{F_i}$ onto image space, and M is the unknown real feature motion that occurred. The chosen objective function (Eq 3.2) is not only intuitive – as it describes the distance between expected projected positions of old features and their corresponding new features –, it provides the maximum likelihood motion estimate [10]. The last remaining issue seems to be the initial parameter – M – values. A straightforward solution is to initialize M as $\tilde{M}_t^f$. However, the use of several perturbed versions of $\tilde{M}_t^f$ as the initial values for M proves to be a very efficient means of drastically increasing robustness to convergence on local minima. For the experiments presented in the Section 4, the tuned LM algorithm developed in this work converges and terminates on average after 3 iterations. This further justifies the use of several initial solutions to increase robustness.

Let M^* be the computed optimal feature motion, the computed camera motion becomes $\breve{M}_t^c = -M^*$. As we do with $\breve{M}_t^i$, we associate a variance to $\breve{M}_t^c$ that we will use later when updating the pose estimate. This variance reflects the performance of the minimization algorithm by relating the number of feature matches N to the total error E obtained by computing Eq 3.2 such that when E is of a similar magnitude as N, $\breve{M}_t^c$ is deemed very "trustworthy" and is given a low variance, whereas as E increase with respect to N, so does $\breve{M}_t^c$'s variance.

3.3 Updating the pose estimate

A pose P is a 6-degree vector of the form $< \vec{P}, \vec{O} >$ where $\vec{P}$ denotes the 3D coordinates of the camera and $\vec{O}$ denotes 3D camera orientation [6]. A naive approach to updating the current pose estimate $P_{t-\Delta t}$ with the measured egomotion $\breve{M}_t$ extracted from the incoming sensor reading would translate $P_{t-\Delta t}.\vec{P}$ by $\breve{M}_t.\vec{T}$ and rotate $P_{t-\Delta t}.\vec{O}$ by $\breve{M}_t.\vec{R}$. However, not only does this naive approach ignore the variance associated to $\breve{M}_t$, it also disregards the predicted egomotion which can be seen in this context as the dual of $\breve{M}_t$. Hence, an additional step is required to merge the measured and predicted motions before the resulting "merged" motion is applied to the pose estimate. This merging is done with a Kalman filter-based approach reminiscent of the one presented earlier. The sole noteworthy detail of this merging is that the duality between measurement and prediction is exploited in the choice of the prediction variance which is set to

$$\tilde{M}_t^c.\sigma = 1 - \breve{M}_t.\sigma$$

4 Evaluation

4.1 Experimental Setup

We tested our approach with realistic synthetic data. Great care was taken to maximize its realism via the noise models detailed below.

4.2 IMU and Camera Noise Models

We situated our approach in the context of usage with relatively inexpensive PCEDs; this implies a substantial level of noise should be tolerated. The noise model is based on a candidate sensor, the *MicroStrain Inertia-Link* ® [12], shown in Figure 4.1. The process of generating a synthetic inertial reading is two-fold. First, we generate a ground-truth reading, i.e. the exact 3D sensor orientation and linear acceleration. Then, we perturb this ideal reading with Gaussian noise such that the resulting noisy reading is within the candidate device's accuracy. We performed a test of this noise model to see if egomotion estimation based solely on synthetic inertial readings would report movement even in a stationary context. This erroneous motion reporting is a common issue in modern inertial navigation systems and is a cause of *drift*. It is often reported that inertial navigation systems can drift by up to one nautical

[6] The origin of this 6D space is the initial pose of the camera

Figure 4.1. *MicroStrain Inertia-Link* ® **IMU.**

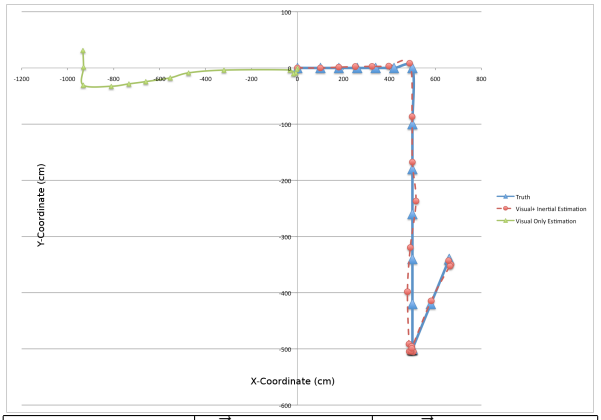
mile in one hour [7]. The test consisted in sending inertial readings to the egomotion estimator at a rate of 50Hz for a simulated time of 1 hour and observing reported motion. The test succeeded and confirmed the validity of the simulated inertial sensor in that the estimated poses did drift over time: after 1 hour, the system reported a measured motion of approximately 0.8 nautical miles.

The process of generating a synthetic visual reading is also two-fold. First, we generate a noiseless reading, i.e. a list of 2D feature points with exact coordinates. Then, we perturb this ideal reading to simulate the noise included in an actual camera shot, i.e. blur, salt-and-pepper and Gaussian noise, by perturbing feature point coordinates and randomly removing features and adding new spurious ones.

4.3 Experiments and Results

This subsection details a variety of tests that demonstrate the introduced approach's ability to cope with "difficult motions" that would leave visual- or inertial-only egomotion estimators at a loss. Furthermore, to establish a basis for comparison with other work, we reproduce the main test from [16] and [17] which claim to produce *optimal* egomotion estimates via batch and online algorithms. The results presented below were obtained using 17 features with varying depths and spanning the 4 quadrants of image space.

Drift Test: As mentioned above, estimating egomotion using only inertial information results in pose estimates that considerably drift over time. However, a sensible requirement of any egomotion system is to report no motion when no motion has occurred. To test the introduced approach's robustness to drift, we sent inertial and visual readings to the egomotion estimator for a simulated time of 1 hour and then observed the reported motion. The result was a positional error of



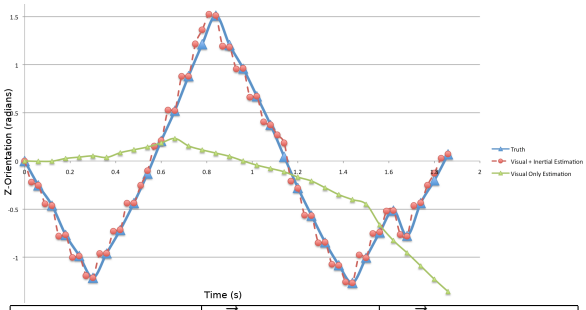
Method	$\vec{O}$ **error** (rad)	$\vec{P}$ **error** (cm)
Visual Only	0.08 / 0.06	1203.57 / 1500.41
Visual+Inertial	0.00 / 0.00	14.79 / 12.73

Figure 4.2. The orientation and position errors are measured as the sum of the errors along each axis. The error columns show the average error before the slash and the median error after the slash.

slightly more than 10 centimeters, as opposed to an error of nearly one nautical mile when using inertial readings alone. A very similar positional error was reported when running the test again using visual readings only. Hence, it appears that though our technique is subject to some drift due its random process underpinnings, it is robust to noisy inertial measurements.

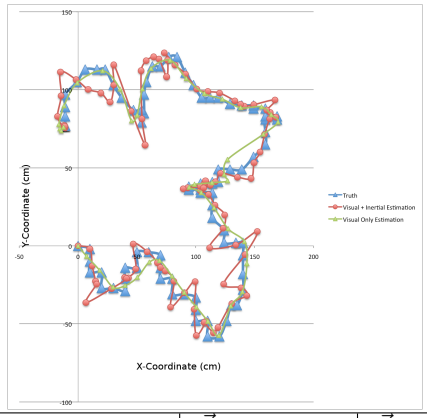
Erratic Translations Test: This test was made up of three very abrupt translations in three different directions. Visual and inertial readings were sent to the estimator regularly throughout the motion. Figure 4.2 shows the real and estimated pose sequences along with tabulated results and demonstrates the system's ability to robustly estimate egomotion by fusing visual and inertial information in a context where using only visual information results in complete failure of feature tracking.

Erratic Rotations Test: This test was made up of a series of intertwined clockwise and counter-clockwise rotations of up to 140°. Much like the previous test, it demonstrates the approach's robustness to motions that cripple the visual tracker. The main difference is that this test demonstrates proper tracking of rotations rather than translations. Figure 4.3 shows the real and estimated Z orientation along with tabulated results.



Method	$\vec{O}$ **error** (rad)	$\vec{P}$ **error** (cm)
Visual Only	0.59 / 0.59	220.08 / 180.39
Visual+Inertial	0.02 / 0.00	5.90 / 4.58

Figure 4.3. Combining visual and inertial readings enables near perfect estimation whereas the visual only tracker is at a loss.



Method	$\vec{O}$ **error** (rad)	$\vec{P}$ **error** (cm)
Visual Only	*0.00 / 0.00*	*4.65 / 4.53*
Visual Only B[16]	0.45 / ??	15.1 / ??
Visual Only B[17]	0.49 / ??	19.0 / ??
Visual+Inertial	*0.00 / 0.00*	*8.08 / 6.30*
Visual+Inertial B[16]	0.07 / ??	4.3 / ??
Visual+Inertial B[17]	0.15 / ??	3.4 / ??
Visual+Inertial O[17]	0.28 / ??	6.4 / ??

Figure 4.4. Our results our compared to those reported in [16] and [17].

"Clover Leaf" Test [16] [17]: This test meant to compare our approach's performance with that of the approach from [16] and [17] which claims to produce *optimal* egomotion estimates via batch and online algorithms. Hence, this test described a motion very similar to the one used in the evaluations of those works. Figure 4.4 shows the real and estimated pose sequences along with the tabulated results of our own experiments and those obtained in [16] and [17] on a similar albeit less erratic motion.

The depicted motions for the *Erratic rotations* and *Erratic translations* tests are so abrupt that the visual-only estimator fails completely while the merged visual and inertial system performs exceedingly well. As for test 4, the obtained results are very encouraging in that they are either extremely close or slightly better that those produced by a so-called *optimal* estimator. Furthermore, it appears that the built-in robustness of the visual tracking algorithms allow our visual-only estimator to perform very well on so-called "difficult" motions. For ease of visualization, the previous tests avoid 6-degree motions. However, our technique is fully capable of properly sensing and estimating 6-degree motions. Testing did however uncover one weakness of our approach which is a direct consequence of the geometric foundations of the camera model: the performance of the minimization algorithm is severely affected by the number of features and their disposition in space. A higher – but not too high – number of features and a greater – but not too great – coverage of the 4 quadrants of image space yields even better results than those reported above. However, considerably reducing or increasing the number of features and their coverage of image space can cause performance to decrease drastically.

5 Future Work

Several features and improvements need to be studied and incorporated into our approach to make it more robust and practical. A brief overview of these follows.

Obtaining new feature points as yet unseen portions of the world become visible. We are currently investigating a technique that trades-off speed and accuracy to acquire new features and estimate their 3D coordinates.

Though no situations have produced the need to do so yet, we could increase the overall system's robustness to noise by keeping parallel pose estimate structures – versus the unique data structure currently used – when several motions are near equally likely. The intuition is that, as time goes by, some of the pose sequences would become less likely and could be pruned. This is of course another speed versus accuracy trade-off.

We could make several performance and robustness enhancements to the visual tracker. For instance, merging feature detection and tracking would avoid looking for "new" features where no old features are expected to be. Also, we could eliminate false matches in a few ways, one of which is to make several passes through the entire feature tracking and measured motion solving processes to remove outliers and iteratively improve the quality of the predicted motion. Furthermore, an outlier removal step would also have the advantage of increasing robustness to movement in the observed scene.

We are currently in the process of acquiring real data to test our approach. This will entail the integration of a feature detector in the approach. Noteworthy candidates are SIFT [11] and FAST [14].

Finally, we will also explore the precise benefits and computational costs of using edges to increase robustness to camera motion blur. Interesting edge-based egomotion estimation approaches are [8], [13] and [9].

6 Conclusion

Due to their complementary strengths, inertial and visual readings are prime candidates to be used together to estimate egomotion. Our approach set out to exploit this complimentary nature to efficiently and robustly estimate motion within a noisy, computationally limited and unconstrained environment. Our results have shown that despite targeting a platform that prohibits exhaustiveness, we can produce very precise and robust egomotion estimates.

References

[1] G. Alenya, C. Torras, and E. Martinez. Fusing visual and inertial sensing to recover robot egomotion. *Journal of Robotic Systems*, 21:23–32, 2004.

[2] A. Blake and M. Isard. *Active Contours*. Springer, first edition, 2000.

[3] A. R. Bruss and B. K. P. Horn. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21(1):3–20, 1983.

[4] P. Corke, J. Lobo, and J. Dias. An introduction to inertial and visual sensing. *The International Journal of Robotics Research*, 26:519–535, 2007.

[5] W. Flenniken, J. Wall, and D. Bevly. Characterization of various imu error sources and the effect on navigation performance. In *ION GNSS*, 2005.

[6] J. Gluckman and S. Nayar. Ego-motion and omnidirectional cameras. In *Sixth International Conference on Computer Vision*, 1998.

[7] A. King. Inertial navigation – forty years of evolution. *GEC Review*, 13:140–149, 1998.

[8] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. In *British Machine Vision Conference*, pages 787–796, 2002.

[9] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *10th European Conference on Computer Vision*, pages 802–815, 2008.

[10] R. Kumar, A. Tirumalai, and R. Jain. A non-linear optimization algorithm for the estimation of structure and motion parameters. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1989.

[11] D. G. Lowe. Object recognition from local scale-invariant features. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[12] MicroStrain. Inertia-link :: Microstrain inertial measurement unit. `http://www.microstrain.com/inertia-link.aspx?gclid=CKi1g53HpJQCFQNHFQodejwPtg`.

[13] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *International Conference on Computer Vision*, pages 1508–1515. Springer, 2005.

[14] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006.

[15] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *IEEE International Conference on Robotics and Automation*, pages 4326–4333, 2002.

[16] D. Strelow and S. Singh. Optimal motion estimation from visual and inertial measurements. In *Sixth IEEE Workshop on Applications of Computer Vision*, 2002.

[17] D. Strelow and S. Singh. Online motion estimation from image and inertial measurements. In *Workshop on Integration of Vision and Inertial Sensors*, 2003.

[18] N. Sunderhauf, K. Konolige, S. Lacroix, and P. Protzel. Visual odometry using sparse bundle adjustment on an autonomous outdoor vehicle. *Autonome Mobile Systeme*, 19:157–163, 2005.

[19] T. Y. Tian, C. Tomasi, and D. J. Heeger. Comparison of approaches to egomotion computation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 315–320, 1996.

[20] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[21] R. F. Vassallo, J. Santos-Victor, and H. J. Schneebeli. A general approach for egomotion estimation with omnidirectional images. In *IEEE Workshop on Omnidirectional Vision*, pages 97–103, 2002.