Hand Waving Away Scale

Anonymous ECCV submission

Paper ID 1135

Abstract. This paper presents a novel solution to the metric reconstruction of objects using any smart device equipped with a camera and an inertial measurement unit (IMU). We propose a batch, vision centric approach which only uses the IMU to estimate the metric scale of a scene reconstructed by any algorithm with Structure from Motion like (SfM) output. IMUs have a rich history of being combined with monocular vision for robotic navigation and odometry applications. These IMUs require sophisticated and quite expensive hardware rigs to perform well. IMUs in smart devices, however, are chosen for enhancing interactivity - a task which is more forgiving to noise in the measurements. We anticipate, however, that the ubiquity of these "noisy" IMUs makes them increasingly useful in modern computer vision algorithms. Indeed, we show in this work how an IMU from a smart device can help a face tracker to measure pupil distance, and an SfM algorithm to measure the metric size of objects. We also identify motions that produce better results, and develop a heuristic for estimating, in real-time, when enough data has been collected for an accurate scale estimation.

Keywords: Smart devices, IMU, metric, 3D reconstruction

1 Introduction

Obtaining a metric reconstruction of the 3D world is a problem that has largely been ignored by the computer vision community when using monocular or multiple uncalibrated cameras. This ignorance is well founded, Structure from Motion (SfM) [1] dictates that a 3D object/scene can be reconstructed up to an ambiguity in scale. The vision world, however, is changing. Smart devices (phones, tablets, etc.) are low cost, ubiquitous and packaged with more than just a monocular camera for sensing the world. Even digital cameras are being bundled with a plethora of sensors such as GPS (global positioning system), light intensity, and IMUs (intertial measurement units).

The idea of combining measurements of an IMU and a monocular camera to make metric sense of the world has been well explored by the robotics community [2–7]. Traditionally, however, the community has focused on odometry and navigation which requires accurate and as a consequence expensive IMUs while using vision largely in a periphery manner. IMUs on modern smart devices, in contrast, are used primarily to obtain coarse measurement of the forces being applied to the device for the purposes of enhancing user interaction. As a consequence costs can be reduced by selecting noisy, less accurate sensors. In isolation they are largely unsuitable for making metric sense of the world.

In this paper we explore an offline vision centric strategy for obtaining metric reconstructions of the outside world using noisy IMUs commonly found in smart devices.

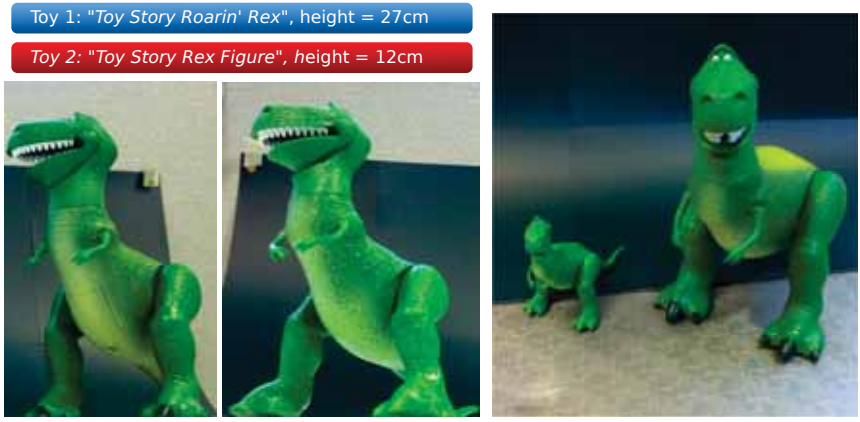


Fig. 1: Scale ambiguitiues can introduce detection ambiguities. These two toys are similar in shape but vary greatly in size. How could a toy detector know the difference if they are not in the same shot or share a common reference?

Specifically, we put forward a strategy for estimating everything about the world using vision except scale. We rely only on the IMU for the scale estimate. The strength of our strategy lies in the realisation that when the entire subject remains in the frame, scale does not change over time. Assuming that IMU noise is largely uncorrelated and there is sufficient motion during the collection of the video, we hypothesise that such an approach should converge eventually towards an accurate scale estimate even in the presence of significant amounts of IMU noise.

Applications in Vision: By enabling existing vision algorithms (operating on IMU enabled digital cameras such as smart devices) to make metric measurements of the world, they can be improved and new applications discovered. Figure 1 demonstrates how the lack of metric scale not only introduces ambiguities in SfM style applications, but in other common tasks in vision such as object detection. For example, a standard object detection algorithm could be employed to detect a toy dinosaur in a visual scene. However, what if the task is not only to detect the type of toy, but to disambiguate between two similar toys that differ only in scale? Unless the shot contains both toys (see right-most image in Figure 1) or some other reference object, there would be no simple way visually to separate them. Similarly, a pedestrian detection algorithm could know that a doll is not a person. In biometric applications an extremely useful biometric trait for separating people is the scale of the head (e.g. pupil distance), which goes largely unused by current facial recognition algorithms. Alternatively, a 3D scan of an object using a smart device could be 3D printed to precise dimensions using our approach combined with SfM algorithms.

Contributions: In this paper we make the following contributions.

– We propose an elegant batch-style objective for recovering scale with a noisy IMU and monocular vision. A strength of our approach is that it can be seamlessly inte-

grated with any existing vision algorithm that is able to obtain accurate SfM style camera motion matrices, and the 3D structure of the object of interest up to an ambiguity in scale and reference frame. (Section 3.2)
- A novel strategy for aligning video and IMU input on a smart device using gravity. Most[1] smart devices do not synchronise the IMU and video. If the IMU and video inputs are not sufficiently aligned, we demonstrate that the scale estimate accuracy in practice is severely degraded. A strength of our alignment strategy, which takes advantage of gravity rather than removing it, is that it is independent of device and operating system. (Section 3.4)
- Finally, we propose an approach for ascertaining in real-time when enough device motion has occurred to ensure an accurate measure of scale can be obtained through our method. (Section 3.5)

We demonstrate the utility of our approach for obtaining metric scale across a number of visual tasks such as obtaining a metric reconstruction of a chessboard, estimating pupil distance, and obtaining a metric 3D reconstruction of a toy dinosaur. This is the first work of its kind, to our knowledge, to get such accurate (in all our experiments we achieved scale estimates within $1-2\%$ of ground-truth) metric reconstructions using a canonical smart device's monocular camera and IMU.

2 Related Work

2.1 Non-IMU Methods

There are ways to obtain a metric understanding of the world using monocular vision on a smart device that do not require an IMU. They all pivot on the idea of obtaining a metric measurement of something already observed by the vision algorithm and propogating the corresponding scale. There are a number of apps [8, 9] which achieve this using vision. However, they all require some kind of external reference in order to estimate the metric scale factor of the vision, such as credit cards or knowing height of the camera from the ground (assuming the ground is flat).

2.2 IMU Methods

Online Methods: Our paper in many ways overlaps with existing robotics literature for combining monocular camera and IMU inputs. It differs in that many of these algorithms are focussed on navigation and odometry, and so the algorithms must execute in real-time.

Works by Jones et al. [6], Nützi et al. [2], Weiss et al. [3], and Li et al. [7] all show how the camera motion of any visual SLAM (simultaneous localisation and mapping) algorithm can be fused with accelerometer and gyroscope measurements using a Kalman Filter. The IMU measurements (at 100Hz or more) are integrated to estimate motion and errors are corrected each time the SLAM is updated (20Hz).

Weiss et al. [3] take the idea a step further by automatically detecting failures in the SLAM output and use only the IMU until the SLAM algorithm recovers. The objectives

[1] We tested our proposed approach on both iOS and Android smart devices, neither of which provided global timestamps for the video input.

of Weiss' work are similar to ours in that their implementation is modular to any SLAM algorithm that provides position and orientation, and they assess the quality of the scale estimate in their results.

Li et al. [7] account for rolling-shutter distortion that occurs in low quality cameras. Unlike the above mentioned methods they do apply their approach to a smart device. However, they still focus mainly on navigation, and the odometry. SLAM feature tracking, and sensor fusion are all tightly integrated and nonmodular.

Offline Methods: Offline methods are advantageous, as they do not require close integration with the vision algorithm when computing scale. They can often times give more accurate estimates of scale, as they attempt to solve the problem using all the data at the same time (i.e. in batch) unlike online methods. Offline methods have a further advantage in that they allow a "plug and play" strategy for incorporating various object-centric vision algorithms (e.g. face trackers, chessboard trackers, etc.) with little to no modification.

Jung and Taylor [4] present an offline method to fuse IMU and camera data in batch using spline approximations, with only a handful of camera frames being used to estimate the camera trajectory. Like previous online works the focus of this work was on recovering odometry. We believe one of the core motivations for the use of splines was to reduce computational requirements. Splines allow the data to be broken up into "epochs", reducing the dimensionality of the final problem, however this also reduces the resolution. This causes problems if the camera is moving too quickly.

Skoglund et al. [5] propose another offline method that enhances an SfM problem by including IMU data in the objective. The camera and IMU are high quality and secured to a custom rig. The IMU motion is first integrated so that its trajectory can be compared with that of the camera's. Unlike with smart devices, the high quality of sensors allows this to be done without introducing too many compounding errors. An estimation of scale is obtained but is not the central focus of the work.

Tanskanen et al. [10] demostrate a pipeline for real-time metric 3D reconstruction, however they never discuss the accuracy of the metric scale estimation. Finite segments of large motions are detected heuristically and estimates for the displacement measured by the IMU and by the camera are compared. An estimation of the scene scale is obtained by executing a batch least squares which minimises the difference between these two displacement estimates. This is accurate enough to help increase the robustness of the 3D reconstruction but it is unclear how precise the dimensions of the final model are.

3 Recovery of Scale

Using SfM (Structure from Motion) algorithms, or algorithms tailored for specific objects (such as chessboards, faces, cars) we can determine the 3D camera pose and scene accurately up to scale. This section describes a batch, vision centric approach which, other than the camera, only uses a smart device's IMU to estimate the metric scale. All that is required from the vision algorithm is the position of the center of the camera, and its orientation in the scene.

3.1 In One Dimension

The scale factor from vision units to real units is time invariant and so with the correct assumptions about noise, an estimation of its value should converge to the correct answer with more and more data. Let us consider the trivial one dimensional case

$$\begin{aligned} \arg\min_{s} \ & \eta\{s\nabla^2\mathbf{p}_V - \mathbf{D}\mathbf{a}_I\} \\ \text{s.t.} \ & s > 0, \end{aligned} \tag{1}$$

where $\mathbf{p}_V$ is the position vector containing samples across time of the camera in vision units, $\mathbf{a}_I$ is the metric acceleration measured by the IMU, ∇^2 is the discrete temporal double deriviative operator, and $\mathbf{D}$ is a convolutional matrix that antialiases and downsamples the IMU data. Scale by definition must be greater than zero, we include this here to remain general to the method used to solve the problem. $\eta\{\}$ is some penalty function; the choice of $\eta\{\}$ depends on the noise of the sensor data. This could commonly be the $\ell 2$-norm^2, however we remain agnostic to entertain other noise assumptions. Downsampling is necessary since IMUs and cameras on smart devices typically record data at 100 Hz and 30 Hz, respectively. Blurring before downsampling reduces the effects of aliasing.

The approach here allows us to be modular with the way camera motion is obtained and allows us to compare accelerations rather than positions. This idea differs from work such as [11] and [10] which incorporates the scale estimation into an SfM algorithm by comparing the position of the camera with the position integrated from IMU data (prone to drift and compounding errors).

Equation 1 makes the following assumptions: (i) measurement noise is unbiased and Gaussian (in the case that $\eta\{\}$ is $\ell 2$-norm^2), (ii) the IMU only measures acceleration from motion, not gravity, (iii) the IMU and camera samples are temporally aligned and have equal spacing. In reality, this is not the case. First, IMUs (typically found in smart devices) have a measurement bias that is mostly variant to temperature [12]. Second, acceleration due to gravity is omnipresent. However, most smart device APIs provide a "linear acceleration" which has gravity removed. Third, smart device APIs provide a global timestamp for IMU data but timestamps on video frames are relative to the begining of the video, and so we cannot trivially obtain their alignment. These timestamps do reveal, however, that the spacing between samples in all cases is uniform with little variance. Subsection 3.3 describes the method used to temporally align the data.

These facts allow us to modify our assumptions: (i) when used over a period of 1-2 minutes IMU noise is Gaussian and has a constant bias, (ii) the "linear acceleration" provided by device APIs is sufficiently accurate, (iii) the IMU and camera measurements have been temporally aligned and have equal spacing.

For simplicity we let the acceleration of the vision algorithm, $\mathbf{a}_V = \nabla^2\mathbf{p}_V$. Given the modified assumptions we introduce a bias factor into the objective

$$\arg\min_{s,b} \eta\{s\mathbf{a}_V - \mathbf{D}(\mathbf{a}_I - \mathbf{1}b)\} \, . \tag{2}$$

Note we also omit the $s > 0$ constraint from Equation 1 as it unnecessarily complicates the objective. If a solution to s is found that violates this constraint the solution can be immediately discounted.

3.2 In Three Dimensions

In the following subsection we consider the case where the smart device is moving and rotating in 3D space. Most SfM algorithms will return the position and orientation of the camera in scene coordinates, and IMU measurements are in local, body-centric coordinates. To compare them we need to orient the acceleration measured by the camera with that of the IMU. We define the acceleration matrix such that each row is the (x, y, z) acceleration for each video frame

$$\mathbf{A}_V = \begin{pmatrix} a_1^x & a_1^y & a_1^z \\ \vdots & \vdots & \vdots \\ a_F^x & a_F^y & a_F^z \end{pmatrix} = \begin{pmatrix} \Phi_1^\intercal \\ \vdots \\ \Phi_F^\intercal \end{pmatrix} . \tag{3}$$

Then we rotate the vectors in each row to obtain the body-centric acceleration measured by the vision algorithm

$$\hat{\mathbf{A}}_V = \begin{pmatrix} \Phi_1^\intercal \mathbf{R}_1^V \\ \vdots \\ \Phi_F^\intercal \mathbf{R}_F^V \end{pmatrix} \tag{4}$$

where F is the number of video frames, $\mathbf{R}_n^V$ is the orientation of the camera in scene coordinates at the nth video frame.

Similarly to $\mathbf{A}_V$, we form an $N \times 3$ matrix of IMU accelerations, $\mathbf{A}_I$, where N is the number of IMU measurements.

We also need to ensure that IMU measurements are spatially aligned with the camera coordinate frame. Since the camera and IMU are on the same circuit board, this is an orthogonal transformation, $\mathbf{R}_I$, that is determined by the API used by the smart device [13, 14]. We use the rotation to find the IMU acceleration in local camera coordinates.

This leads to the following objective, noting that antialiasing and downsampling have no effect on constant bias

$$\arg\min_{s,\mathbf{b}} \eta\{s \cdot \hat{\mathbf{A}}_V + \mathbf{1} \otimes \mathbf{b}^\intercal - \mathbf{D}\mathbf{A}_I\mathbf{R}_I\} . \tag{5}$$

3.3 Temporal Alignment

Temporal alignment is important for accurate results - Figure 7 shows that scale estimation is not possible without it. Equations 2 and 5 assume that the camera and IMU measurements are temporally aligned. This subsection describes a method to determine the delay between the signals and thus align them for processing.

The optimum alignment between two signals can be found by first calculating their cross-correlation. The cross-correlation is then normalised by dividing each of its elements by the number of elements from the original signals that were used to calculate it. The index of the maximum normalised cross-correlation value is chosen as the delay between the signals.

Before aligning the two signals, an initial estimate of the biases and scale can be obtained using Equation 5. These values can be used to adjust the acceleration signals in order to improve the results of the cross-correlation. The optimisation and alignment are alternated until the alignment converges.

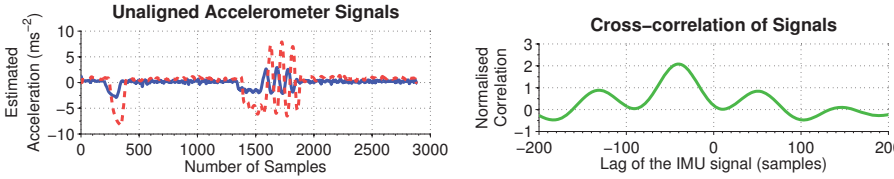


Fig. 2: Showing the result of the normalised cross-correlation of the camera and IMU signals. *Blue-solid line*: camera acceleration scaled by initial solution. *Red-dashed line*: IMU acceleration. The delay that gives the best alignment here is approximately 40 samples.

3.4 Gravity as a Friend

The above method for finding the delay between two signals can struggle with smaller motions when data is particularly noisy. Reintroducing gravity has two advantages: (i) it behaves as an anchor to significantly improve the robustness of the alignment, (ii) allows us to remove the black box gravity estimation built in to smart devices with IMUs.

Instead of comparing the estimated camera acceleration and linear IMU acceleration, we add the gravity vector, $\mathbf{g}$, back into the camera acceleration and compare it with the raw IMU acceleration (which already contains gravity). Before superimposing gravity, it needs to be oriented with the IMU acceleration, much like the vision acceleration

$$\hat{\mathbf{G}} = \begin{pmatrix} \mathbf{g}^\intercal \mathbf{R}_1^V \\ \vdots \\ \mathbf{g}^\intercal \mathbf{R}_F^V \end{pmatrix} . \tag{6}$$

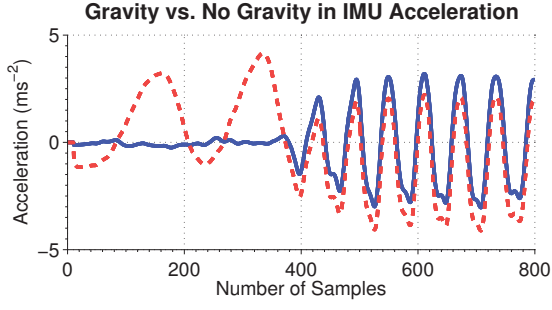


Fig. 3: The large, low frequency motions of rotation through the gravity field helps anchor the temporal alignment. *Blue solid line*: IMU acceleration with gravity removed. *Red dashed line*: raw IMU acceleration measuring gravity.

Since the accelerations are in the camera reference frame the reintroduction of gravity essentially captures the pitch and roll of the smart device. The red dashed line in

Figure 3 shows that the gravity component is of relatively large magnitude and low frequency. This can improve the robustness of the alignment dramatically.

If the alignment of the vision scene with gravity is already known, it can simply be added to the camera acceleration vectors before estimating the scale. However, to keep our method general we extend the above objectives to include the gravity term

$$\underset{s,\mathbf{b},\mathbf{g}}{\arg\min}\, \eta\{s\hat{\mathbf{A}}_V + \mathbf{1} \otimes \mathbf{b}^T + \hat{\mathbf{G}} - \mathbf{D}\mathbf{A}_I\mathbf{R}_I\} \tag{7}$$

where $\mathbf{g}$ is linear in $\hat{\mathbf{G}}$.

Note that Equation 7 does not attempt to constrain gravity to its known constant value. This is addressed by alternating between solving for $\{s, \mathbf{b}\}$ and $\mathbf{g}$ separately where $\mathbf{g}$ is normalised to its known magnitude when solving for $\{s, \mathbf{b}\}$. This is iterated until the scale estimation converges.

3.5 Classifying Useful Data

When recording video and IMU samples offline it is important to know when one has sufficient samples. We classify which parts of the signal are useful by ensuring it contains enough excitation. This is achieved by centering a window at sample, n, and computing the spectrum through short time Fourier analysis. A sample is classified as useful if the amplitude of certain frequencies is above a chosen threshold.

The selection of the frequency range and threshold is investigated in the experiments in Section 4.1. Note that the minimum size of the window is limited by the lowest frequency one wishes to classify as useful.

4 Experiments

In the following experiments, sensor data is collected from iOS and Android devices using custom built applications. The applications record video while logging IMU data at 100Hz to a file. These files are then processed in batch as described in the experiments. For all the experiments, the cameras' intrinsic calibration matrices have been determined beforehand, and the camera is pitched and rolled at the beginning of each sequence to help temporal alignment of sensor data (Section 3.4).

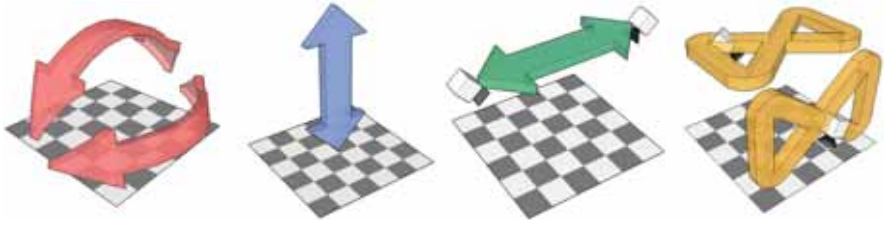
The choice of $\eta\{\}$ depends on the assumptions of the noise in the data. In many cases we obtained good empirical performance with the $\ell 2$-norm^2 penalty (Equation 8. However, we also explored alternate penalty functions such as the grouped-$\ell 1$-norm that are less sensitive to outliers. We obtain camera motion in three different ways: (i) track a chessboard of unknown size, (ii) use pose estimation of a face-tracking algorithm [15], (iii) use the output of an SfM algorithm.

4.1 Chessboard Experiments

On an iPad, we assess the accuracy of the of scale estimation described in Section 3.2 and the types of trajectories that produce the best results. Using a chessboard allows us to be agnostic from objects and obtaining the pose estimation from chessboard corners is well researched. We used OpenCV's *findChessboardCorners* and *solvePnP* functions.

The trajectories in these experiments were chosen in order to test the number of axes that need to be excited, the trajectories that work best, the frequencies that help the most, and the required amplitude of the motions. They can be placed into the four following categories (shown in Figure 4):

(a) Orbit Around: The camera remains the same distance to the centroid of the object while orbiting around,
(b) In and Out: The camera moves linearly toward and away from the object,
(c) Side Ways: The camera moves linearly and parallel to a plane intersecting the object,
(d) Motion 8: The camera follows a figure of 8 shaped trajectory - this can be in or out of plane.



(a) Orbit Around (b) In and Out (c) Side Ways (d) Motion 8

Fig. 4: The above diagrams show the different categories of trajectories. The accuracies of different combinations of these trajectories are assessed. In each case, the camera is is always looking at the subject.

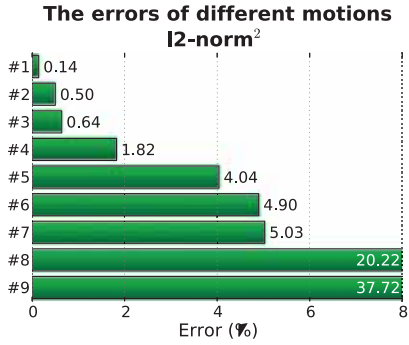
Different sequences of the four trajectories were tested. The use of different penalty functions, and thus different noise assumptions, is also explored. Figure 5 shows the accuracy of the scale estimation when we choose the $\ell 2$-norm^2 (Equation 8). Figure 6 shows the results when we choose the grouped-$\ell 1$-norm (Equation 9). There is an obvious overall improvement when using the grouped-$\ell 1$-norm, suggesting that a Gaussian noise assumption is not strictly observed.

$$\eta_{\ell 2}\{\mathbf{X}\} = \sum_{i=1}^{F} \|\mathbf{x}_i\|_2^2 \tag{8}$$

$$\eta_{\ell 2 \ell 1}\{\mathbf{X}\} = \sum_{i=1}^{F} \|\mathbf{x}_i\|_2 \tag{9}$$

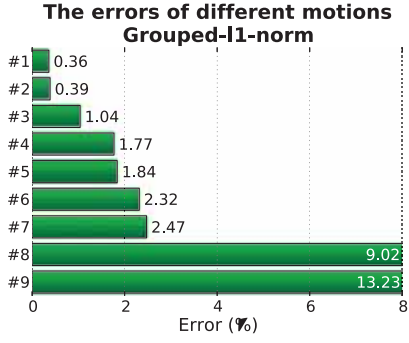
$$\text{where } \mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_F]^\mathsf{T} \tag{10}$$

Both Figures 5 and 6 show that, in general, it is best to excite all axes of the smart device. The most accurate scale estimation was achieved by combination of *In and Out (b)* and *Sideways (c)* motion (along both the x and y axes) and is shown in Figure 8.



# Motions	Frequency (Hz)	Excitation (s) X	Y	Z
1 b + c(X and Y axis)	~1	20	30	45
2 b + c(X and Y axis)	~1.2	35	25	70
3 b + c(X and Y axis)	~0.8	10	7	5
4 b + c(X and Y axis)	~0.7	10	10	10
5 b	~0.75	0	0	160
6 b + c(X and Y axis)	~0.8	5	3	4
7 b + c(X and Y axis)	~1.5	7	6	4
8 a(X and Y axis) + b	0.4-0.8	30	30	47
9 b + d(in plane)	~0.8	50	50	10

Fig. 5: The percentange error in scale estimations for different motions on an iPad. Linear trajectories produce more accurate estimations. Labelled according to Figure 4.



# Motions	Frequency (Hz)	Excitation (s) X	Y	Z
1 b + c(X and Y axis)	~0.8	10	7	5
2 b + c(X and Y axis)	~0.7	10	10	10
3 b + c(X and Y axis)	~0.8	5	3	4
4 b + c(X and Y axis)	~1.5	7	6	4
5 b + c(X and Y axis)	~1	20	30	45
6 b	~0.75	0	0	160
7 b + c(X and Y axis)	~1.2	35	25	70
8 a(X and Y axis) + b	0.4-0.8	30	30	47
9 b + d(in plane)	~0.8	50	50	10

Fig. 6: The same sequences in Figure 5 are used to estimate scale using the group L1 norm. Overall improvement suggests a non-Gaussian noise distribution. Labelled according to Figure 4.

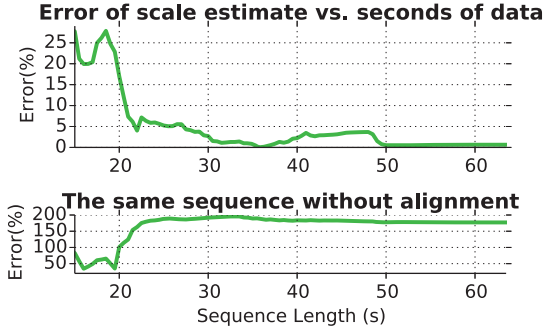


Fig. 7: The scale estimation converges (with the addition of data) to the ground truth over time for $b + c$ motions in all axes. For completeness we also show the error when the camera and IMU signals are not temporally aligned.

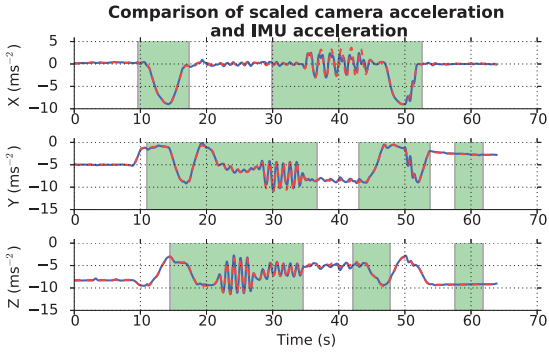


Fig. 8: The sequence *b + c(X,Y)* excites multiple axes which increases the accuracy of scale estimations. *Blue solid line*: scaled camera acceleration. *Red dashed line*: IMU acceleration. For completeness we highlight, in shaded areas, the segments that are classified as useful motions.

Figure 7 shows the estimate of scale as a function of the length of the sequence used. It shows that scale estimate improves with the addition of new data to have an error of less than 2% with just 55 seconds of motion. This method does not focus on odometry, thus removing the risk of divergence that can occur when integrating accelerometer data.

From these observations, we build a real-time heuristic for knowing when enough data has been collected. Upon inspection of the results shown in Figure 5 we can construct the following criteria for sufficiently accurate results: (i) all axes should be excited with (ii) more than 10 seconds of motions of amplitude larger than 2ms^{-2}.

4.2 Measuring Pupil Distance

In this experiment, we test our method's ability to accurately measure the distance between one's pupils with an iPad. Using a facial landmark tracking SDK [15], we can obtain the camera pose relative to the face and locations of facial landmarks (with local variations to match the indiviual). We assume that, for the duration of the sequence, the face keeps the same expression and that the head remains still. To reflect this, the facial landmark tracking SDK was modified to solve for only one expression in the sequence rather than one at each video frame.

Due to the motion blur that the cameras in smart devices are prone to, the pose estimation from the face tracking algorithm can drift and occasionally fail. These errors violate the Gaussian noise assumptions. Improved results were obtained using a grouped-$\ell 1$-norm, but we found in practice even better performance could be obtained through the use of an outlier detection strategy [16] (see Appendix A) in conjunction with the canonical $\ell 2$-norm^2 penalty. It is this strategy we use for the remainder of the experiments in this paper.

Figure 9 shows the deviation of the estimated pupil distance from the true value at selected frames from a video taken on an iPad. With only 68 seconds of data, our

[b]0.237

Fig. 9: 7.0s, ±63.0mm

[b]0.237

Fig. 10: 10.0s, ±51.4mm

[b]0.237

[b]0.237

Fig. 18: 10.0s, ±75.3mm

[b]0.237

Fig. 19: 16.0s, ±64.8mm

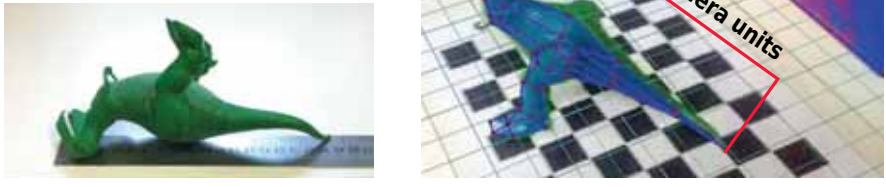
[b]0.237

algorithm can measure pupil distance with sufficient accuracy. Figure 10 shows a similar sequence for a different person. It can be observed that the face tracking, and thus pose estimation, drifts occasionally. In spite of this, the scale estimation is still able to converge over time.

4.3 3D Scanning

In the final experiment, SfM is used to obtain a 3D scan of an object using an Android smart phone. The estimated camera motion from this is used to evaluate the metric scale of the vision coordinates. This is then used to make metric measurements of the virtual object which are compared with those of the original.

The results of these 3D scans can be seen in Figure 11 where a basic model was obtained using VideoTrace [17]. The dimensions estimated by our algorithm are within 1% of the real values. This is sufficiently accurate to help a toy classifier disambiguate the two dinosaur toys shown in (Figure 1).



(a) Measuring the real Rex: 184mm

(b) Measuring the virtual Rex: 0.5653 units = 182.2mm (estimated scale = 322.23)

Fig. 27: The real length of Rex (a) is compared with the length of the 3D reconstruction scaled by our algorithm (b). Sequence recorded on an Android smart phone.

5 Conclusion

This paper has presented a batch technique for obtaining the metric scale of the SfM like output from a vision algorithm using only the IMU on a smart device with less than 2% error. We have made three main contributions that make this possible. First, we realised that by comparing the acceleration of the camera in vision units with the acceleration of the IMU (which we know to be metric), we can find the optimum scale factor to minimise their difference. Second, we have described a method to align sensor measurements which do not have a common timestamp origin (typical on smart device platforms) that uses acceleration from gravity to help anchor the alignment. Finally, we have formed a heuristic to estimate when enough useful data has been collected to make an accurate measurement of metric scale.

References

1. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press, ISBN: 0521540518 (2004)
2. Nützi, G., Weiss, S., Scaramuzza, D., Siegwart, R.: Fusion of IMU and vision for absolute scale estimation in monocular slam. Journal of Intelligent & Robotic Systems **61**(1-4) (2011) 287–299
3. Weiss, S., Achtelik, M.W., Lynen, S., Achtelik, M.C., Kneip, L., Chli, M., Siegwart, R.: Monocular vision for long-term micro aerial vehicle state estimation: A compendium. Journal of Field Robotics **30**(5) (2013) 803–831
4. Jung, S.H., Taylor, C.J.: Camera trajectory estimation using inertial sensor measurements and structure from motion results. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). Volume 2., IEEE (2001) II–732
5. Skoglund, M., Sjanic, Z., Gustafsson, F.: Initialisation and estimation methods for batch optimisation of inertial/visual slam. Technical report, Department of Electrical Engineering Linköpings Universitet (2013)
6. Jones, E., Vedaldi, A., Soatto, S.: Inertial structure from motion with autocalibration. In: Workshop on Dynamical Vision. (2007)
7. Li, M., Kim, B.H., Mourikis, A.I.: Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In: IEEE International Conference on Robotics and Automation (ICRA). (2013) 4712–4719
8. Smart Tools co.: Smart Measure Pro. Google Play Store, http://goo.gl/JDRu5 (2013)
9. Kamens, B.: RulerPhone - Photo Measuring. Apple App Store, http://goo.gl/CRaIOk (2010–2013)
10. Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., Pollefeys, M.: Live metric 3d reconstruction on mobile phones. (2013)
11. Konolige, K., Agrawal, M., Sola, J.: Large-scale visual odometry for rough terrain. In: Robotics Research. Springer (2011) 201–212
12. Aggarwal, P., Syed, Z., Niu, X., El-Sheimy, N.: A standard testing and calibration procedure for low cost mems inertial sensors and units. Journal of navigation **61**(02) (2008) 323–336
13. Android Documentation: SensorEvent - Android Developers. http://goo.gl/fBFBU (2013)
14. iOS Documentation: UIAcceleration Class Reference. http://goo.gl/iwJjKN (2010)
15. Cox, M.J., Nuevo, J.S., Lucey, S.: Deformable model fitting by regularized landmark mean-shift. International Journal of Computer Vision (IJCV) **91**(2) (2011) 200–215
16. Rosner, B.: Percentage points for a generalized ESD many-outlier procedure. Technometrics **25**(2) (1983) 165–172
17. van den Hengel, A., Dick, A., Thormählen, T., Ward, B., Torr, P.H.: Videotrace: rapid interactive scene modelling from video. In: ACM Transactions on Graphics (TOG). Volume 26., ACM (2007) 86

A Outlier Detection

In practice, the output of different tracking algorithms can produce noise which violates the Gaussian noise assumption, Figure 12 shows how tracking errors cause spikes in the acceleration measured by tracking a face. After obtaining an initial estimate of scale, Generalised ESD (extreme Studentised deviate) [16] is used to detect samples whose errors do not follow a Gaussian assumption. These samples are excluded and a new scale estimation is obtained. This method requires only two iterations. Generalised ESD takes two parameters: the sensitivity of outlier detection, α, and an upper bound of the number of outliers to detect, k. In our experiments, the performance of this method was not sensitive to the tuning of these parameters.

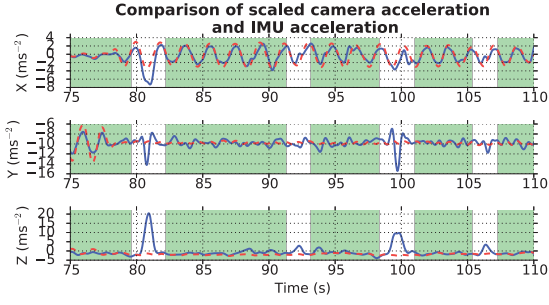


Fig. 28: A comparision of the acceleration measured by the IMU and that estimated by the visual tracking. Outliers violating a Gaussian error assumption have been detected and removed by Generalised ESD. *Blue solid line*: scaled camera acceleration. *Red dashed line*: IMU acceleration.