

SANDIA REPORT

SAND2007-2462

Unlimited Release

Printed January 2007

SLAM Using Camera and IMU Sensors

Fred Rothganger and Maritza Muguira

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>



SLAM Using Camera and IMU Sensors

Fred Rothganger and Maritza Muguira
Cognitive and Exploratory Systems Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1009

Abstract

Visual simultaneous localization and mapping (VSLAM) is the problem of using video input to reconstruct the 3D world and the path of the camera in an “on-line” manner. Since the data is processed in real time, one does not have access to all of the data at once. (Contrast this with structure from motion (SFM), which is usually formulated as an “off-line” process on all the data seen, and is not time dependent.) A VSLAM solution is useful for mobile robot navigation or as an assistant for humans exploring an unknown environment. This report documents the design and implementation of a VSLAM system that consists of a small inertial measurement unit (IMU) and camera. The approach is based on a modified Extended Kalman Filter. This research was performed under a Laboratory Directed Research and Development (LDRD) effort.

1 Introduction

The Simultaneous Localization and Mapping (SLAM) problem usually starts with an unknown environment where one hopes to generate a map and localize within the map. In addition, there is uncertainty in the sensor data that must be used. The development of SLAM is generally credited to Smith and Cheeseman [1]. They were interested in how the relative position estimate of a landmark or the robot degraded by the motion of the robot. They also wanted to determine how the relative position estimate of the landmark or robot was upgraded by new measurements. They framed this mapping and localization problem in probabilistic terms and introduced Monte Carlo Localization and the Extended Kalman Filter for robot localization. Advantages of the Extended Kalman Filter are that the map becomes fully correlated in the limit and it is an analytical approach. Disadvantages include the Gaussian assumptions, computational complexity, and problems with false data association. An Extended Kalman Filter is a recursive Bayesian Filter. Particle filters are another type of recursive Bayesian Filter. One of the advantages of the particle filter approach is the ability to model non-gaussian probability density functions. In fact, any probability density function may be approximated by a set of weighted particles [2]. This paper describes an approach based on a modified Extended Kalman Filter. The steps in this algorithm are

Step 1 Prediction

Step 2 Update from visual features

Step 3 Update from inertial measurement unit

Step 4 Addition of new landmarks

Step 5 Bootstrapping the map

Step 6 Bundle adjustment

These steps are described in more detail in each of the subsequent sections. Test results are briefly presented in Section 8.

2 Prediction

1. $X = [q \ T \ v \ w \ L_1 \ \dots \ L_n]^T$ is the current state of the system. $q = [r \ i \ j \ k]^T$ is the quaternion that rotates coordinates in the camera frame into the world frame, and $T = [x \ y \ z]^T$ translates camera coordinates into world coordinates. (T effectively gives the position of the camera in the world.) v is the velocity vector in m/s , and w is the Rodriguez vector giving the axis and rate of rotation in rad/s . $L = [x \ y \ z]^T$ are the coordinates of the stationary landmarks in the world. All translational coordinates are in meters.
2. $X \leftarrow f(X, V, W, t)$ is the state transition function, where V and W are the unknown errors in velocity

and rotation rate respectively, and t is the amount of time since the last state update. In explicit form:

$$X = f(X, V, W, t) = \begin{bmatrix} M(Q((w + W) \cdot t))q \\ T + (v + V) \cdot t \\ v + V \\ w + W \\ L_1 \\ \vdots \\ L_n \end{bmatrix} \quad (1)$$

where $Q()$ is a function that converts a Rodriguez vector into a quaternion, and $M()$ is a function that converts a quaternion into a quaternion multiplication matrix. Note that the landmarks remain constant. That is, only the camera moves relative to the world frame. In practice, we compute the predicted state vector as $X \leftarrow f(X, 0, 0, t)$ since V and W are unknown. For that reason, we will omit them wherever they are always zero in the equations below.

We will need the following Jacobians of f :

$$\frac{\partial f}{\partial X} = \begin{bmatrix} M(Q(wt)) & 0 & 0 & \frac{\partial}{\partial w} M(Q(wt))q & 0 \\ 0 & I_3 & tI_3 & 0 & 0 \\ 0 & 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & 0 & I_{3n} \end{bmatrix} \quad (2)$$

$$\frac{\partial f}{\partial V} = \begin{bmatrix} 0 \\ tI_3 \\ I_3 \\ 0 \\ \vdots \end{bmatrix} \quad (3)$$

$$\frac{\partial f}{\partial W} = \begin{bmatrix} \frac{\partial}{\partial W} M(Q((w+W) \cdot t))q \\ 0 \\ 0 \\ I_3 \\ 0 \\ \vdots \end{bmatrix} \quad (4)$$

Let $wt = [x \ y \ z]^T$, $a = |wt|$ and $[b \ c \ d \ e]^T = Q(wt)$. We have

$$\frac{\partial}{\partial w} M(Q(wt))q = \frac{\partial Mq}{\partial(Q/|Q|)} \frac{\partial(Q/|Q|)}{\partial Q} \frac{\partial Q}{\partial(wt)} \frac{\partial(wt)}{\partial w} \quad (5)$$

where

$$\frac{\partial Mq}{\partial(Q/|Q|)} = \begin{bmatrix} r & -i & -j & -k \\ i & r & k & -j \\ j & -k & r & i \\ k & j & -i & r \end{bmatrix} \quad (6)$$

$$\frac{\partial(Q/|Q|)}{\partial Q} = \begin{bmatrix} 1-b^2 & -bc & -bd & -be \\ -bc & 1-c^2 & -cd & -ce \\ -bd & -cd & 1-d^2 & -de \\ -be & -ce & -de & 1-e^2 \end{bmatrix}, \text{ and} \quad (7)$$

$$\frac{\partial Q}{\partial(wt)} = \frac{1}{a} \begin{bmatrix} \frac{-x}{2} \sin \frac{a}{2} & \frac{-y}{2} \sin \frac{a}{2} & \frac{-z}{2} \sin \frac{a}{2} \\ \frac{x^2}{2a} \cos \frac{a}{2} + (1 - \frac{x^2}{a^2}) \sin \frac{a}{2} & (\frac{1}{2a} \cos \frac{a}{2} - \frac{1}{a^2} \sin \frac{a}{2})xy & (\frac{1}{2a} \cos \frac{a}{2} - \frac{1}{a^2} \sin \frac{a}{2})xz \\ (\frac{1}{2a} \cos \frac{a}{2} - \frac{1}{a^2} \sin \frac{a}{2})xy & \frac{x^2}{2a} \cos \frac{a}{2} + (1 - \frac{x^2}{a^2}) \sin \frac{a}{2} & (\frac{1}{2a} \cos \frac{a}{2} - \frac{1}{a^2} \sin \frac{a}{2})yz \\ (\frac{1}{2a} \cos \frac{a}{2} - \frac{1}{a^2} \sin \frac{a}{2})xz & (\frac{1}{2a} \cos \frac{a}{2} - \frac{1}{a^2} \sin \frac{a}{2})yz & \frac{x^2}{2a} \cos \frac{a}{2} + (1 - \frac{x^2}{a^2}) \sin \frac{a}{2} \end{bmatrix} \quad (8)$$

We also have $\frac{\partial}{\partial W} M(Q((w+W) \cdot t))q = \frac{\partial}{\partial w} M(Q(wt))q$ since $\frac{\partial((w+W) \cdot t)}{\partial W} = t = \frac{\partial(wt)}{\partial w}$, and the derivative is taken around $W = 0$.

3. $P \leftarrow \frac{\partial f}{\partial X} P \frac{\partial f}{\partial X}^T + \frac{\partial f}{\partial V} \Sigma_V \frac{\partial f}{\partial V}^T + \frac{\partial f}{\partial W} \Sigma_W \frac{\partial f}{\partial W}^T$ is the predicted process error covariance matrix. P has the block form

$$P = \begin{bmatrix} \Sigma_{[q \ T \ v \ w]} & 0 & \cdots & 0 \\ 0 & \Sigma_{L_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \Sigma_{L_n} \end{bmatrix} \quad (9)$$

$\Sigma_{[q \ T \ v \ w]}$ is a 13×13 covariance matrix that describes the uncertainty in the location orientation of the camera, along with the uncertainty in its rotation rate and velocity. Each Σ_L is a 3×3 covariance matrix that describes the uncertainty in the location of the landmark. Note that we are forcing the off-diagonals of the covariance to zero. This is technically incorrect, and yields only an approximation of the optimal Kalman filter. We are throwing away two kinds of information:

- (a) How to revise the position of a landmark when the current position of the sensor rig is revised, and vice-versa. This information extends to all landmarks, not just those currently seen by the camera.
- (b) How to revise the position of a landmark when another landmark is revised.

What we retain is:

- (a) The dependency between landmarks currently seen by the camera due to uncertainty in the position of the camera. The information is generated from $\Sigma_{[q \ T \ v \ w]}$ when constructing the Kalman gain matrix.
- (b) The mobility of landmarks, that is, the uncertainty of their locations.

We can compensate for the lost information using a numerical process other than Kalman filtering, such as resection-intersection. The advantage of this approach is that data management is simpler and costs only $O(n)$ rather than $O(n^2)$ space. Note that throwing away all dependence between landmarks is the basis of FastSLAM, where it is justified by claiming full certainty about camera positions within a given “particle”.

$\Sigma_V = \sigma_V^2 I_3$ and $\Sigma_W = \sigma_W^2 I_3$ are the estimated covariances of the errors V and W respectively. Note that we never possess V and W explicitly, but instead model them implicitly via their covariances. We pretend that all parameters are independent, so the covariances only have values on the diagonal, and we pretend that all parameters have equal amounts of error. The values σ_V and σ_W give estimates of how far off the velocity and rotation rate are at a given moment. Reasonable numbers are $0.1m/s$ (1% of the maximum human land-speed of about $10m/s$) and $0.01rad/s$ (about $0.1rpm$).

3 Update from Visual Features

1. $h(X)$ is the sensor function, which maps the state vector to expected positions of feature points in the image. It has the form

$$h(X) = \begin{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_1 \\ \vdots \\ \begin{bmatrix} u \\ v \end{bmatrix}_m \end{bmatrix} \quad (10)$$

The form of h varies from one image to the next due to the varying subsets of visible landmarks. It outputs a vector containing a pair of rows for each of m visible landmarks in the current image. The predicted image positions $[u \ v]_i^T$ are computed by perspective projection. Let R be the rotation matrix associated with q (the current orientation of the camera):

$$R = \begin{bmatrix} 1 - 2j^2 - 2k^2 & 2ij - 2rk & 2ik + 2rj \\ 2ij + 2rk & 1 - 2i^2 - 2k^2 & 2jk - 2ri \\ 2ik - 2rj & 2jk + 2ri & 1 - 2i^2 - 2j^2 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (11)$$

For a given feature i and its associated landmark $L_{(i)}$, the sensor function is

$$h(X)_i = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{c} \begin{bmatrix} a \\ b \end{bmatrix} \quad (12)$$

where

$$[a \ b \ c]^T = R^T(L_{(i)} - T) \quad (13)$$

We assume that the feature coordinates have already been multiplied by the camera's intrinsic matrix normalized to the unit plane. To simplify the expression below, let $[x \ y \ z]^T = L_{(i)} - T$ so we have $[a \ b \ c]^T = R^T[x \ y \ z]^T$.

We will need the Jacobian of h to compute the Kalman update. The Jacobian H has a pair of rows for each visible landmark in the current image, and columns equal to the size of X . Its block form is

$$H = \begin{bmatrix} \frac{\partial h_1}{\partial q} & \frac{\partial h_1}{\partial T} & 0 & 0 & \frac{\partial h_1}{\partial L_{(1)}} & 0 & 0 & 0 \\ \frac{\partial h_2}{\partial q} & \frac{\partial h_2}{\partial T} & 0 & 0 & 0 & \frac{\partial h_2}{\partial L_{(2)}} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & 0 & 0 & \ddots & 0 \\ \frac{\partial h_m}{\partial q} & \frac{\partial h_m}{\partial T} & 0 & 0 & 0 & 0 & 0 & \frac{\partial h_m}{\partial L_{(m)}} \end{bmatrix} \quad (14)$$

Any column associated with a non-visible landmark contains all zeros. The definitions of the sub-blocks in H are

$$\frac{\partial h_i}{\partial q} = \frac{\partial h_i}{\partial R} \frac{\partial R}{\partial q} = \begin{bmatrix} \frac{x}{c} & \frac{y}{c} & \frac{z}{c} & 0 & 0 & 0 & \frac{-ax}{c^2} & \frac{-ay}{c^2} & \frac{-az}{c^2} \\ 0 & 0 & 0 & \frac{x}{c} & \frac{y}{c} & \frac{z}{c} & \frac{-bx}{c^2} & \frac{-by}{c^2} & \frac{-bz}{c^2} \end{bmatrix} \frac{\partial R}{\partial q}, \quad (15)$$

$$\frac{\partial h_i}{\partial T} = \frac{1}{c^2} \begin{bmatrix} ar_{13} & -cr_{11} & ar_{23} & -cr_{21} & ar_{33} & -cr_{31} \\ br_{13} & -cr_{12} & br_{23} & -cr_{22} & br_{33} & -cr_{32} \end{bmatrix} \text{ and} \quad (16)$$

$$\frac{\partial h_i}{\partial L_{(i)}} = -\frac{\partial h_i}{\partial T}. \quad (17)$$

We also have

$$\frac{\partial R}{\partial q} = \frac{\partial R}{\partial(q/|q|)} \frac{\partial(q/|q|)}{\partial q} = \begin{bmatrix} 0 & 0 & -4j & -4k \\ 2k & 2j & 2i & 2r \\ -2j & 2k & -2r & 2i \\ -2k & 2j & 2i & -2r \\ 0 & -4i & 0 & -4k \\ 2i & 2r & 2k & 2j \\ 2j & 2k & 2r & 2i \\ -2i & -2r & 2k & 2j \\ 0 & -4i & -4j & 0 \end{bmatrix} \begin{bmatrix} 1-r^2 & -ri & -rj & -rk \\ -ri & 1-i^2 & -ij & -ik \\ -rj & -ij & 1-j^2 & -jk \\ -rk & -ik & -jk & 1-k^2 \end{bmatrix} \quad (18)$$

based on the observation that q must always have a norm of 1, so any perturbation in one of its parameters produces changes in the other three.

2. $K = PH^T(HPH^T + S)^{-1}$ is the Kalman gain, where H is the Jacobian of the sensor function h with respect to the state vector X and S is the covariance of the sensor error. Note that this is equivalent to solving $(HPH^T + S)K^T = HP$ for K^T , since P and S are symmetric. The sensor covariance S is defined as $\sigma_C^2 I_{2m}$, where σ_C gives the uncertainty in pixel location of the features. A possible choice is the radius of the pixel (in the unit plane). This assumes the only source of uncertainty is noise in the retina of the camera. Other sources to consider are errors in tracking points between images. These can be detected as discontinuities in the trajectories of various parameters associated with a given point. It can also be detected after the fact by measuring the reprojection error.
3. $X \leftarrow X + K(z_c - h(X))$ updates the estimate with image measurement z_c , which consists of a stack of (u, v) positions for the feature points. After computing X , the quaternion q must be normalized to unit length.
4. $P \leftarrow P - KHP$ updates the error covariance matrix.

4 Update from IMU

The Xsens IMU provides acceleration, angular velocity, and magnetic field measurements. In addition the Xsens device driver computes a filtered orientation value, which we also treat as a sensor. The sensor value from the IMU is then

$$z_{imu} = \begin{bmatrix} a \\ \omega \\ \beta \\ \theta \end{bmatrix} \quad (19)$$

where a contains the three components of linear acceleration, ω is a Rodriguez vector with the same semantics as w , β contains the three components of the magnetic field, and θ is a 3×3 rotation matrix with the same semantics as R .

1. A Kalman update step on these values would be mostly vacuous, since there is no reasonable way to predict them. Instead we integrate them and place the results directly in X . This is essentially a prediction step that uses information from the IMU in addition to / instead of the current state:

$$X \leftarrow e(X, z_{imu}, V, A, \Theta, t) = \begin{bmatrix} M(Q((\omega + \Theta) \cdot t))q \\ T + (v + V) \cdot t + \frac{1}{2}(a + A) \cdot t^2 \\ (v + V) + (a + A) \cdot t \\ \omega + \Theta \\ L_1 \\ \vdots \\ L_n \end{bmatrix}. \quad (20)$$

The error vector V is defined as above, the vector A is the error in the IMU acceleration measurement, and the Rodriguez vector Θ is the error in the IMU angular velocity measurement. Note that we use the angular velocity rather than simply assigning the quaternion equivalent of θ to q because the IMU measures changes in angle very well, but is subject to error in the absolute angle due to magnetic disturbances.

In practice we use a slightly more sophisticated model when there is sufficient radial acceleration (that

is, acceleration perpendicular to the direction of travel):

$$\begin{aligned}
x &= \frac{v}{|v|}, \quad y = \frac{z \times v}{|z \times v|}, \quad z = \frac{x \times a}{|x \times a|} \\
a_{\parallel} &= a \cdot x, \quad a_{\perp} = a \cdot y \\
l &= \frac{v^T v}{a_{\perp}}, \quad \alpha = \frac{|v|t}{l} + \frac{a_{\parallel} t^2}{2l} \\
T &\leftarrow T + x(l \sin \alpha) + y(l - l \cos \alpha) \\
v &\leftarrow (|v| + a_{\parallel} t)(x \cos \alpha + y \sin \alpha)
\end{aligned} \tag{21}$$

First, we calculate orthonormal basis vectors x and y in the plane of the angular motion, with x pointing in the direction of forward (tangential) motion and y pointing in the direction of centripetal (radial) motion. We then compute the components of acceleration that will contribute to the tangential and radial motion. Next we determine the radius and length of the arc traced out in the given time. The new position T is the endpoint of this arc, and the final velocity vector v is the exit speed pointing in the direction of the tangent at the end of the arc. In the limit (for very small radial acceleration components) this reduces to the simple formula given for T and v in (20).

2. $P \leftarrow \frac{\partial e}{\partial X} P \frac{\partial e}{\partial X}^T + \frac{\partial e}{\partial V} \Sigma_V \frac{\partial e}{\partial V}^T + \frac{\partial e}{\partial A} \Sigma_A \frac{\partial e}{\partial A}^T + \frac{\partial e}{\partial \Theta} \Sigma_{\Theta} \frac{\partial e}{\partial \Theta}^T$ is the update to the state covariance associated with the IMU-based prediction. To compute it, we need the following Jacobians:

$$\frac{\partial e}{\partial X} = \begin{bmatrix} M(Q(wt)) & 0 & 0 & 0 & 0 \\ 0 & I_3 & tI_3 & 0 & 0 \\ 0 & 0 & I_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{3n} \end{bmatrix} \tag{22}$$

$$\frac{\partial e}{\partial V} = \begin{bmatrix} 0 \\ tI_3 \\ I_3 \\ 0 \\ \vdots \end{bmatrix}, \tag{23}$$

$$\frac{\partial e}{\partial A} = \begin{bmatrix} 0 \\ \frac{t^2}{2} I_3 \\ t I_3 \\ 0 \\ \vdots \end{bmatrix}, \quad (24)$$

and

$$\frac{\partial e}{\partial \Theta} = \begin{bmatrix} \frac{\partial}{\partial W} M(Q((\omega + \Theta) \cdot t))q \\ 0 \\ 0 \\ I_3 \\ 0 \\ \vdots \end{bmatrix} \quad (25)$$

with

$$\left. \frac{\partial}{\partial \Theta} M(Q((\omega + \Theta) \cdot t))q \right|_{\Theta=0} = \frac{\partial(Mq)}{\partial(Q/|Q|)} \frac{\partial(Q/|Q|)}{\partial Q} \frac{\partial Q}{\partial((\omega + \Theta) \cdot t)} \frac{((\omega + \Theta) \cdot t)}{\partial \Theta} \Big|_{\Theta=0} = \frac{\partial(Mq)}{\partial(Q/|Q|)} \frac{\partial(Q/|Q|)}{\partial Q} \frac{\partial Q}{\partial(\omega t)} t \quad (26)$$

Let $\Sigma_A = \sigma_A^2 I_3$ and $\Sigma_\Theta = \sigma_\Theta^2 I_3$. The value σ_A gives the expected amount of error in the IMU accelerometer measurements. The Xsens documentation places this at $0.01m/s^2$, although experiment shows it to be around $0.03m/s^2$. The value σ_Θ gives the expected amount of error in the IMU angular velocity measurements. The Xsens documentation places this at $0.01rad/s$.

5 Adding new Landmarks

1. As we start tracking the same feature point in multiple 2D images, we can extract the 3D position of the feature. The standard projection equation is

$$\begin{bmatrix} p \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} R^T & -R^T T \end{bmatrix} \begin{bmatrix} L \\ 1 \end{bmatrix}, \quad (27)$$

where z is the value of the third row of the product on the right hand side. Let

$$\begin{bmatrix} A & B \\ a & b \end{bmatrix} = \begin{bmatrix} R^T & -R^T T \end{bmatrix}, \quad (28)$$

where A is the first two rows of the rotation matrix R^T and B is the first two rows of the translation vector $-R^T T$. The projection equation can be expressed as the function

$$p = f(L) = \frac{AL + B}{aL + b}, \quad (29)$$

which can be rearranged to solve for L in terms of the known camera positions and 2D image measurements:

$$(A - pa)L = pb - B. \quad (30)$$

Given two or more views of L , we can solve for it by minimizing the squared error in a set of such equations. Alternately, we can solve for multiple landmarks and camera positions simultaneously using non-linear least squares on a set of standard projection equations. (This setup is typically referred to as projective structure from motion, and solving it with some non-linear method is called “bundle adjustment”.)

2. After calculating the 3D world position, we need to add the process covariance for the new landmark: Σ_L . Since L is computed from several measurements, each appearing in a different image, we define a modified sensor function

$$g(L, [q \ T]_1, \dots, [q \ T]_m) \quad (31)$$

which expresses the projection of L into each of the m images where it appears. Since v and w have no direct effect on perspective projection, they are not included. Also, we are abusing the notation $[q \ T]_i$ to mean $[q^T \ T^T]_i^T$. The associated Jacobian is

$$G = \begin{bmatrix} \frac{\partial h_1}{\partial L} & \frac{\partial h_1}{\partial [q \ T]_1} & 0 & \dots & 0 \\ \frac{\partial h_2}{\partial L} & 0 & \frac{\partial h_2}{\partial [q \ T]_2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial L} & 0 & 0 & \dots & \frac{\partial h_m}{\partial [q \ T]_m} \end{bmatrix}, \quad (32)$$

computed using the same basic blocks described in Section 3. The landmark covariance is

$$\Sigma_L = \frac{1}{m} \left[\frac{\partial L}{\partial [q \ T]} \Sigma_{[q \ T]} \frac{\partial L}{\partial [q \ T]}^T + \frac{\partial h}{\partial L}^T S \frac{\partial h}{\partial L} \right], \quad (33)$$

where $[q \ T]$ stands for the combined parameters of all the camera positions where L was seen. The factor $1/m$ compensates for the fact the $\Sigma_{[q \ T]}$ and S have zero off-diagonals. Effectively, the matrix products shown above amount to sums of m smaller matrix products, each adding the covariance associated with one measurement. If the off-diagonals were present, they would adjust for the relationships between the measurements. Instead, we approximate this by taking the average.

We must find the Jacobian of L with respect to the camera parameters. One possible solution is to assert that the value of the sensor function h does not vary when any of $[q \ T]$ or L vary, and determine how these variables relate to each other:

$$\frac{\partial h}{\partial [L \ q \ T]} \Delta [L \ q \ T] = 0 \quad (34)$$

$$\frac{\partial h}{\partial L} \Delta L = - \frac{\partial h}{\partial [q \ T]} \Delta [q \ T] \quad (35)$$

$$\Delta L = - \frac{\partial h}{\partial L} + \frac{\partial h}{\partial [q \ T]} \Delta [q \ T] \quad (36)$$

$$\frac{\partial L}{\partial [q \ T]} = - \frac{\partial h}{\partial L} + \frac{\partial h}{\partial [q \ T]} \quad (37)$$

At this point in the process, we can directly compute S as the reprojection error rather than estimating it. That is, we set the diagonal elements of S to the difference between g and the actual image measurements. We do not allow this value to go below the pixel quantizing error (0.5 pixels, adjusted by radial distortion).

6 Bootstrapping the Map

When the system starts up, there are a couple of options for initializing the first few landmarks. One choice is to rely on the IMU for camera positions and simply triangulate the landmarks as described in Section 5. Unfortunately, we do not know the initial velocity, and the only available way to estimate it is from the landmarks, creating a circular dependency. Another choice, which resolves the circular dependency, is to use a calibration pattern as the starting point. A third choice, which does not require any special affordances from the environment, is to simultaneously estimate the camera positions and landmarks from a set of image measurements, using factorization to solve the structure from motion (SFM) problem. This is an off-line technique because it requires all the data to be present before running, in contrast to Kalman filtering, which works through the data sequentially. However, we can collect measurements from the first few images and process them very quickly before moving into Kalman filtering.

Here is a quick summary of the method [Tomasi and Kanade]: As a simplification, we use an affine setup. Each camera position has only 6 parameters, essentially the first two rows of its rotation matrix. All coordinates are relative to their respective centroids, so the setup includes no translation vectors. Rather, these are inferred and added in later. We assemble n measurements from m images into a single matrix

$$D = \begin{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_{11} & \dots & \begin{bmatrix} u \\ v \end{bmatrix}_{1n} \\ \vdots & \ddots & \vdots \\ \begin{bmatrix} u \\ v \end{bmatrix}_{m1} & \dots & \begin{bmatrix} u \\ v \end{bmatrix}_{mn} \end{bmatrix} = \begin{bmatrix} A_1 \\ \vdots \\ A_m \end{bmatrix} \begin{bmatrix} L_1 & \dots & L_n \end{bmatrix} = MN \quad (38)$$

which is the product of the landmark positions and the truncated rotation matrices for each camera position. Because of its structure, D has rank 3. We apply a singular value decomposition (SVD) to D , and use only

the first three singular values and their associated eigenvectors to construct a set of camera matrices and landmarks:

$$D = U\Sigma V^T = (U\sqrt{\Sigma})(\sqrt{\Sigma}V^T) = \hat{M}\hat{N}. \quad (39)$$

The SVD produces a model that is generally separated from the “true” Euclidean one by an affine transformation Q : $\hat{M}Q Q^{-1}\hat{N} = MN$. We correct (“upgrade”) the model using the value of Q that brings \hat{M} closest to a set of truncated rotation matrices with scale (that is, where each camera matrix is a pair of orthogonal vectors with equal norm). Specifically, we solve the least squares problem

$$\begin{bmatrix} a_1^T Q Q^T a_2 \\ a_1^T Q Q^T a_1 - a_2^T Q Q^T a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (40)$$

over all m camera matrices, where a_1 and a_2 are the first and second rows of a given matrix A . We rotate the upgraded set of camera matrices to best align with the orientations measured by the IMU. This rotation is the solution X to the least squares problem $XR = \theta$ over all the camera positions, where R is the camera rotation and θ is the IMU rotation.

To build a full set of camera parameters, it is necessary to add back the offsets to the centroids of the various point clouds. We place the first camera position at the origin, and arbitrarily place the centroid of the landmarks 1 meter in front of it (that is, in its unit plane). The scales and orientations of the remaining camera matrices determine their positions relative to the first one. The resulting model generally does not match the real world, but it provides a reasonable starting point which can then be refined by bundle adjustment.

Finally, for the sake of the Kalman filter, we assign a covariance to each camera position and landmark. A method for computing landmark covariance from reprojection errors is given in Section 5. A method for computing camera covariance appears in Section 7.

7 Bundle Adjustment

After each image update, it is useful to propagate information from new image measurements to improve estimates of previous camera positions and of the landmarks. Normally, the Kalman filter will adjust all the landmarks via their relationships in the state covariance matrix P . If we kept all the previous camera positions in the state X , they would be updated as well. However, we have sparsified P by throwing away the off-diagonal blocks, and we don’t keep any but the current camera position. Instead, we use a technique called “resection-intersection” or “alternation”, in which each camera or landmark is re-estimated from the current best values of the others.

To enable efficient operation, we only re-estimate a camera position or landmark when its error exceeds some threshold. Specifically, for each image (and associated camera position) we keep track of the average reprojection error over all of the features seen in it. Likewise, for each landmark we keep track of the average reprojection error over all images where it appears. If, for example, a landmark moves, then we must update the average reprojection error of each camera position that saw the landmark.

Section 5 gave a method for estimating the position of a landmark from several image measurements. Whenever a landmark’s average reprojection error exceeds the threshold, we re-estimate it with all currently available measurements and then update its covariance and reprojection errors.

The method for updating a camera position and is very similar to the Kalman filter for position. The key difference is that we only evaluate the update for the camera position itself and ignore those parts of the calculation that exclusively affect the landmarks. The method for updating camera position covariance is a little different, in that it is computed directly from reprojection error rather than a Kalman update.

Specifically, the covariance for the camera position of an image that contains n measurements (projected landmarks) is

$$\Sigma_{[q \ T]} = \frac{1}{n} \left[\frac{\partial[q \ T]}{\partial L} \Sigma_L \frac{\partial[q \ T]}{\partial L}^T + \frac{\partial h}{\partial[q \ T]}^T S \frac{\partial h}{\partial[q \ T]} \right], \quad (41)$$

where L stands for the combined parameters of all the landmarks seen by the camera. The reasoning behind (41) is similar to that behind (33). We divide by n to compensate for the lack of off-diagonals in Σ_L and S . The Jacobian relating $[q \ T]$ to L is

$$\frac{\partial[q \ T]}{\partial L} = -\frac{\partial h}{\partial[q \ T]}^+ \frac{\partial h}{\partial L} \quad (42)$$

8 Implementation Results

Prototype hardware was developed using a small inertial measurement unit (IMU) mounted to a camera. A picture of the test hardware is shown in Figure 1. The algorithm described in the previous section was implemented on a personal computer (pc). The algorithm in action is shown in Figure 2. The dots in the computer screen in Figure 2 represent the algorithm’s estimate of the location of the book shown in the picture.

References

- [1] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1987.
- [2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: The MIT Press, 2005.

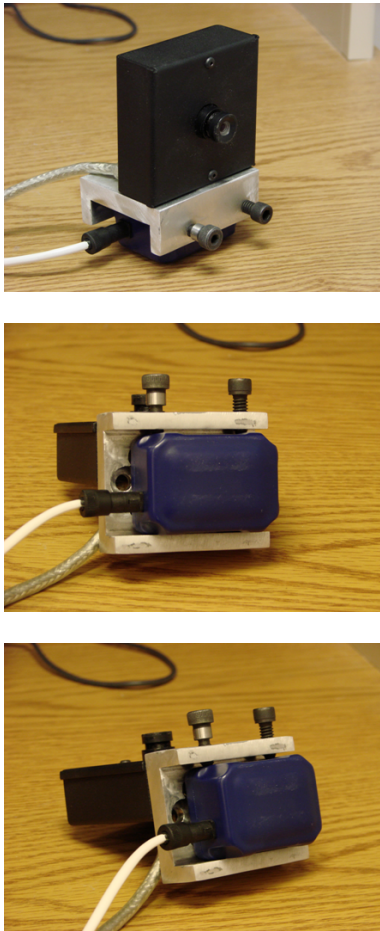


Figure 1: Camera and IMU Hardware

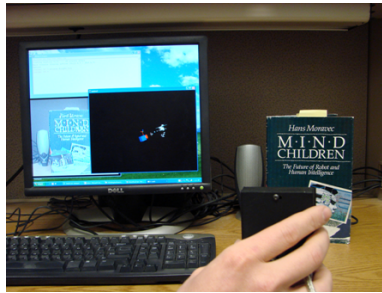
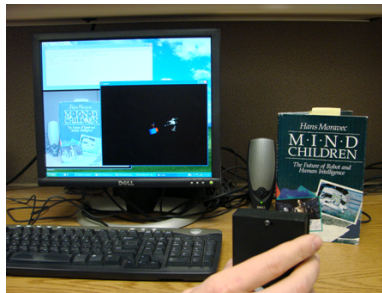


Figure 2: SLAM Software Running on a Personal Computer

Distribution

1	MS1104	Rush Robinett, 6330
1	MS1002	Philip Heermann, 6470
5	MS1003	John Feddema, 6473
1	MS1003	Ray Byrne, 6473
10	MS1009	Fred Rothganger, 6341
1	MS1007	Larry Shipers, 6471
1	MS1125	Phil Bennett, 6472
1	MS1188	John Wagner, 6341
2	MS9018	Central Technical Files, 8944
2	MS0899	Technical Library, 4536