Chapter 3

Vision-based SLAM with Shi-Tomasi Features

3.1 Overview of the System

In this section, we give the overview of our proposed ST-SLAM system. We show the design of the entire system in Fig. 3.1. We assume that a robot exploring an unknown environment has two types of sensors: a robot motion sensor and a measurement sensor that observes objects in the environment. The former is related to the *motion model* of the robot and the latter is related to the *sensor model*. We assume that rough estimates of robot's 6DoF motion is directly available due to some motion sensors (such as wheel encoders and gyro sensors). Our sensor model is completely based on vision. We use a trinocular stereo camera rig as a vision-based sensor. A trinocular image set captured by the rig goes through a series of computational processes (we call it the *sensor pipeline*) to obtain the sparse geometric landmarks of our choice: 3D points.

In the sensor pipeline, we first extract image points from each trinocular image using the Shi-Tomasi corner point detector. We then establish three-way point correspondences across the trinocular images based on the epipolar constraints derived from stereo calibration of the trinocular rig. Finally, we reconstruct 3D points by triangulating the Shi-Tomasi points with three-way correspondences.

For motion and map estimation, we use the FastSLAM algorithm, one of an efficient variants of RBPF. We adapt our motion and sensor models into FastSLAM and supply 6DoF odometry and 3D points as input data. FastSLAM recursively estimates robot pose history and the current environment map.

In the following sections, we explain the FastSLAM algorithm and our sensor model in detail.

3.2 FastSLAM

FastSLAM [1] is an elegant solution to the SLAM problem that maintains a full posterior over possible robot paths (as opposed to a maximum a posteriori estimate) using the RBPF [22]. In this research we adapt Thrun et al.'s "FastSLAM 1.0" [1] algorithm to the vision-based SLAM problem.

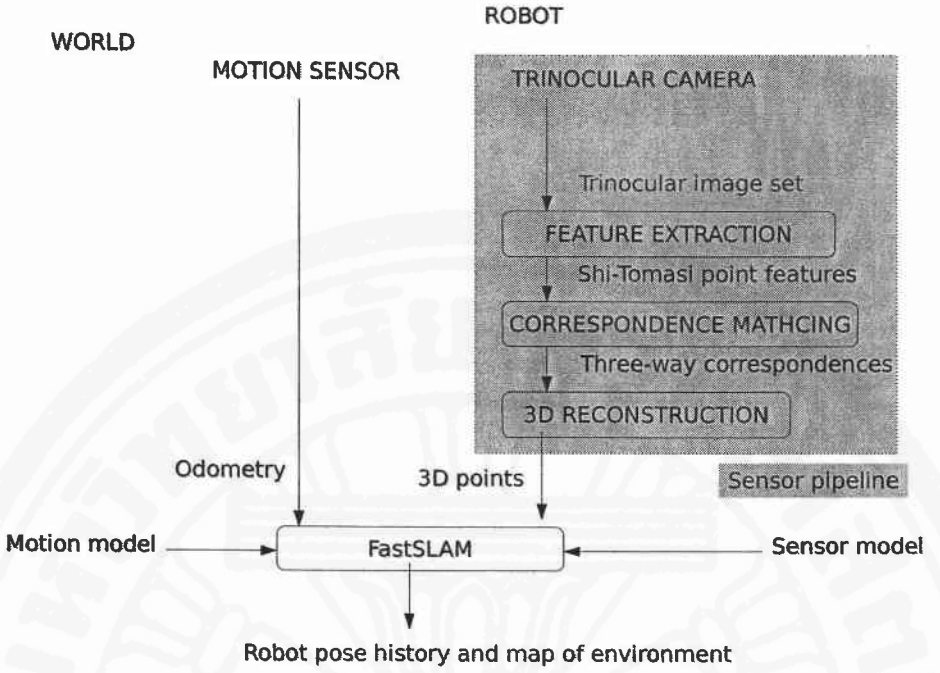


Figure 3.1 Overview of the ST-SLAM system.

3.2.1 The SLAM Posterior

The goal of SLAM problem is to recover the robot's pose at time t, $\boldsymbol{s}_t$ and a map containing a set of landmarks Θ_t from a set of sensor observations $\boldsymbol{z}_{0:t}$, a set of robot actions $\boldsymbol{u}_{1:t}$ and a set of data association assumptions $n_{1:t}$. Most SLAM problems are formulated as a recursive estimation of a probability distribution over the momentary robot pose $\boldsymbol{s}_t$ and the map Θ_t:

$$p(\boldsymbol{s}_t, \Theta_t \mid \boldsymbol{z}_{0:t}, \boldsymbol{u}_{1:t}, n_{1:t}). \tag{3.1}$$

Without loss of generality, we assume that sensor measurements and robot actions alternate each other. As such, data becomes available in the order

$$\boldsymbol{z}_0, \boldsymbol{u}_1, \boldsymbol{z}_1, \ldots, \boldsymbol{u}_i, \boldsymbol{z}_i, \ldots, \boldsymbol{u}_t, \boldsymbol{z}_t. \tag{3.2}$$

Following Montemerlo et al. [1], for brevity of mathematical formulation, we furthermore assume that only one landmark is observed in each sensor measurement, which makes it possible to write sensor observation at time t as a single vector $\boldsymbol{z}_t$ and its data association as a scalar n_t. Note, however, that the result obtained in this section applies to the case of sensor measurement of multiple landmarks. We lift this restriction when we develop our sensor model that handles multiple observations later.

We can recursively estimate the posterior (3.1) based on a Bayes filter equation (see

section A.2 for a derivation):

$$p(\boldsymbol{s}_t, \Theta \mid \boldsymbol{z}_{0:t}, \boldsymbol{u}_{1:t}, n_{1:t}) = \eta\, p(\boldsymbol{z}_t \mid \boldsymbol{s}_t, \boldsymbol{\theta}_{n_t}, n_t) \int p(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{u}_t)\, p(\boldsymbol{s}_{t-1}, \Theta \mid \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t-1}, n_{1:t-1})\, d\boldsymbol{s}_{t-1}, \tag{3.3}$$

where η is a normalization constant, $\boldsymbol{\theta}_{n_t}$ is a landmark that is associated with observation $\boldsymbol{z}_t$ through correspondence n_t, and Θ is a non-dynamic environment with respect to time t. In this thesis, we only focus on mapping static environments with no moving objects, hence we use Θ to denote an environment regardless of the time step of our interest. $p(\boldsymbol{z}_t \mid \boldsymbol{s}_t, \boldsymbol{\theta}_{n_t}, n_t)$ and $p(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{u}_t)$ are referred to as *sensor model* and *motion model* , respectively. We write the sensor model as

$$\boldsymbol{z}_t = \boldsymbol{g}(\boldsymbol{\theta}_{n_t}, \boldsymbol{s}_t) + \boldsymbol{\epsilon}_t, \tag{3.4}$$

where $\boldsymbol{g}$ is a deterministic function to calculate observed landmark position in robot-relative coordinates from an observed landmark and current robot pose, and $\boldsymbol{\epsilon}_t$ is a random variable representing noise of sensor measurement. On the other hand, we write the motion model as

$$\boldsymbol{s}_t = \boldsymbol{h}(\boldsymbol{s}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{\delta}_t, \tag{3.5}$$

where $\boldsymbol{h}$ is a deterministic function to calculate robot pose $\boldsymbol{s}_t$ from the one at one time step earlier and current action, and $\boldsymbol{\delta}_t$ is a random variable representing noise of motion estimation. To make the calculation of the filter equation (3.3) tractable, we assume that $\boldsymbol{\epsilon}_t$ and $\boldsymbol{\delta}_t$ are Gaussian noise. When both $\boldsymbol{g}$ and $\boldsymbol{h}$ are linear functions, (3.3) is equivalent to the Kalman filter that we can easily calculate. Otherwise, we approximate nonlinear function(s) to the first order in order to reduce (3.3) to the extended Kalman filter.

Although estimating the posterior (3.3) suffices to solve the SLAM problem, it is possible to formulate the SLAM problem using a more elaborate posterior that can be also estimated by a Bayes filter:

$$p(\boldsymbol{s}_{0:t}, \Theta \mid \boldsymbol{z}_{0:t}, \boldsymbol{u}_{1:t}, n_{1:t}). \tag{3.6}$$

The posterior (3.6) is a posterior over the robot's path from time 0 to time t and an environment Θ conditioned on a set of sensor observations, a set of action controls, and a set of landmark associations. The filter equation for calculating this posterior (see section A.3 for a derivation) is given by

$$p(\boldsymbol{s}_{0:t}, \Theta \mid \boldsymbol{z}_{0:t}, \boldsymbol{u}_{1:t}, n_{1:t}) = \eta\, p(\boldsymbol{z}_t \mid \boldsymbol{s}_t, \boldsymbol{\theta}_{n_t}, n_t)\, p(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{u}_t)\, p(\boldsymbol{s}_{0:t-1}, \Theta \mid \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t-1}, n_{1:t-1}). \tag{3.7}$$

If the the sensor model and the motion model are defined, we can calculate the posterior (3.6) base on this filter equation. Since we estimate robot's path rather than the momentary pose, the posterior's dimension grows as the time t. This additional complexity, however, does not discourage calculating the posterior (3.6) since keeping track of previous robot poses does not require much computational cost. The true motivation of introducing the posterior (3.6) over the original posterior (3.1) is that landmarks $\{\boldsymbol{\theta}_n\}, n = 1, \ldots, N$, where N is the total number of landmarks in an environment, become conditionally independent by adopting the posterior (3.6) over the robot path, which brings many theoretical advantages in estimating the SLAM posterior as discussed in the next section.

3.2.2 Factoring the SLAM Posterior

Montemerlo et al. [1] observe that to obtain an efficient estimation algorithm for the posterior (3.6), we can write it (see section A.4 for details) using Bayes' rule as

$$p(s_{0:t}, \Theta \mid z_{0:t}, u_{1:t}, n_{1:t}) = p(s_{0:t} \mid z_{0:t}, u_{1:t}, n_{1:t})\, p(\Theta \mid s_{0:t}, z_{0:t}, n_{1:t}), \tag{3.8}$$

where the path posterior $p(s_{0:t} \mid z_{0:t}, u_{1:t}, n_{1:t})$ and the map posterior $p(\Theta \mid s_{0:t}, z_{0:t}, n_{1:t})$ are separated. The map posterior is conditioned on the robot path $s_{0:t}$. Our strategy is that we obtain an estimate of the path $s_{0:t}$ from the path posterior and, based on the path, we estimate the landmarks Θ in the environment. The conditional dependence of the map posterior on the path $s_{0:t}$ is important, since this implies that all landmarks $\{\theta_n\}, n = 1, \ldots, N$ become independent. In other words, given a path $s_{0:t}$, observations $z_{0:t}$ and data associations $n_{1:t}$, the estimate of a particular landmark θ_1 tells nothing about the other landmark θ_2. This fact allows us to write the map posterior in a factorized form (see section A.4 for a mathematical derivation):

$$p(\Theta \mid s_{0:t}, z_{0:t}, n_{1:t}) = \prod_{n=1}^{N} p(\theta_n \mid s_{0:t}, z_{0:t}, n_{1:t}). \tag{3.9}$$

From (3.8) and (3.9), we finally obtain the SLAM posterior decomposed into $N + 1$ independent estimators:

$$p(s_{0:t}, \Theta \mid z_{0:t}, u_{1:t}, n_{1:t}) = p(s_{0:t} \mid z_{0:t}, u_{1:t}, n_{1:t}) \prod_{n=1}^{N} p(\theta_n \mid s_{0:t}, z_{0:t}, n_{1:t}). \tag{3.10}$$

This factorization leads to the design of the RBPF [22]. In RBPF, the posterior distribution over possible robot paths is represented by a set of samples or *particles*, where each particle at time t represents one possible robot path up to time t, one possible series of data association assumptions for the sensor measurements up to time t, and a stochastic landmark map [18] based on those assumptions. Since each particle represents a particular robot path and a particular series of data association decisions up to time t, the observed landmarks are conditionally independent as (3.10). The assumption of a particular robot path and particular series of data associations allows a representation of the map that is linear in the number of landmarks (the classical stochastic map is quadratic in the number of landmarks due to the correlations introduced by uncertain robot positions). So the map can be represented simply as a list of landmark estimates with associated uncertainties. Therefore, we represent the SLAM posterior $p(s_{0:t}, \Theta \mid z_{0:t}, u_{1:t}, n_{1:t})$ by a set of M_t particles $\{S_t^{[m]}, m \in 1, \ldots, M_t\}$, where each m-th particle $S_t^{[m]}$ is written as

$$S_t^{[m]} = \left\langle s_{0:t}^{[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \ldots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\rangle. \tag{3.11}$$

Here we represent the estimate of n-th landmark using a Gaussian distribution with mean $\mu_{n,t}^{[m]}$ and covariance $\Sigma_{n,t}^{[m]}$.

3.2.3 FastSLAM with Unknown Data Association

FastSLAM 1.0 uses sequential importance resampling, also known in the computer vision literature as the "condensation" algorithm [26]. The filter algorithm generates,

for each time step t, a particle set $\{S_t^{[m]}, m \in 1, \ldots, M_t\}$ representing the posterior $p(\boldsymbol{s}_{0:t}, \Theta \mid \boldsymbol{z}_{0:t}, \boldsymbol{u}_{1:t}, n_{1:t})$, from the previous particle set $\{S_{t-1}^{[m]}, m \in 1, \ldots, M_{t-1}\}$. This process of updating the particle set comprises mainly four steps in order:

- Extend the path posterior by sampling new robot poses.
- Establish data association.
- Update landmarks in stochastic maps.
- Resample based on importance factor.

Our approach is identical to planar FastSLAM 1.0 [1] except that we use a six degree of freedom motion model $p(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{u}_t)$ and a 3D point sensor model $p(\boldsymbol{z}_t \mid \boldsymbol{s}_t, \boldsymbol{\theta}_{n_t}, n_t)$ using landmarks derived from Shi-Tomasi features on a trinocular stereo vision rig.

In the following subsections, we explain each step of the FastSLAM 1.0 algorithm.

Extend the path posterior by sampling new robot poses

At each time t, for each particle m, we sample a new pose $\boldsymbol{s}_t^{[m]}$ from a *proposal distribution*. The proposal distribution could be any distribution that approximates the full joint posterior (3.6). It is also important that we can actually sample from that distribution. In FastSLAM 1.0, we use the motion model itself $p(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}, \boldsymbol{u}_t)$ as the proposal distribution:

$$\boldsymbol{s}_t^{[m]} \sim p(\boldsymbol{s}_t \mid \boldsymbol{s}_{t-1}^{[m]}, \boldsymbol{u}_t). \tag{3.12}$$

Establish data association

Assuming the sampled position $\boldsymbol{s}_t$, for the observed landmark $\boldsymbol{z}_t$, we find the most likely correspondence $\hat{n}_t^{[m]}$ with a stored landmark, i.e.

$$\hat{n}_t^{[m]} = \arg\max_{n_t} p(\boldsymbol{z}_t \mid n_t, \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t}^{[m]}, \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t}). \tag{3.13}$$

This method is called the *maximum likelihood data association*. The probability that is maximized in (3.13) is called the *likelihood* of the measurement $\boldsymbol{z}_t$. The likelihood is calculated as follows:

$$\begin{aligned} & p(\boldsymbol{z}_t \mid n_t, \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t}^{[m]}, \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t}) \\ = & \int p(\boldsymbol{z}_t \mid \boldsymbol{\theta}_{n_t}, n_t, \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t}^{[m]}, \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t})\, p(\boldsymbol{\theta}_{n_t} \mid n_t, \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t}^{[m]}, \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t}) d\boldsymbol{\theta}_{n_t} \\ = & \int p(\boldsymbol{z}_t \mid \boldsymbol{\theta}_{n_t}, n_t, \boldsymbol{s}_t^{[m]})\, p(\boldsymbol{\theta}_{n_t} \mid \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t-1}^{[m]}, \boldsymbol{z}_{0:t-1}) d\boldsymbol{\theta}_{n_t}, \end{aligned} \tag{3.14}$$

where the two factors inside the integral are known Gaussian distributions,

$$p(\boldsymbol{z}_t \mid \boldsymbol{\theta}_{n_t}, n_t, \boldsymbol{s}_t^{[m]}) \sim \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{g}(\boldsymbol{\theta}_{n_t}, \boldsymbol{s}_t^{[m]}), \mathrm{R}_t), \tag{3.15}$$

$$p(\boldsymbol{\theta}_{n_t} \mid \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t-1}^{[m]}, \boldsymbol{z}_{0:t-1}) \sim \mathcal{N}(\boldsymbol{\theta}_{n_t}; \boldsymbol{\mu}_{n_t,t-1}^{[m]}, \Sigma_{n_t,t-1}^{[m]}). \tag{3.16}$$

$\mathtt{R}_t$ is the covariance matrix when we approximate the sensor model (3.4) by Gaussian error model. Finally, linearizing $\boldsymbol{g}$ with respect to $\boldsymbol{\theta}_{n_t}$ renders the Gaussian convolution (3.14) in closed form:

$$\begin{aligned} & p(\boldsymbol{z}_t \mid n_t, \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t}^{[m]}, \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t}) \\ = & \left|2\pi \mathtt{Q}_t^{[m]}\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t^{[m]}\right)^T \left(\mathtt{Q}_t^{[m]}\right)^{-1} \left(\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t^{[m]}\right)\right\}, \end{aligned} \tag{3.17}$$

where

$$\mathtt{Q}_t^{[m]} = \mathtt{G}_t^{[m]} \Sigma_{n_t,t-1}^{[m]} \mathtt{G}_t^{[m]T} + \mathtt{R}_t, \tag{3.18}$$

$$\mathtt{G}_t^{[m]} = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\theta}_{n_t}}(\boldsymbol{\mu}_{n_t,t-1}^{[m]}, \boldsymbol{s}_t^{[m]}), \tag{3.19}$$

$$\hat{\boldsymbol{z}}_t^{[m]} = \boldsymbol{g}(\boldsymbol{\mu}_{n_t,t-1}^{[m]}, \boldsymbol{s}_t^{[m]}). \tag{3.20}$$

Data association is done for each particle m, so that the resultant particle set accommodates different data association assumptions per particle. This flexibility makes the particle filtering techniques robust to incorrect data correspondences that lead to serious map inconsistency seen in EKF-based algorithms.

Update landmarks in stochastic maps

In this step, we update the particle's map with $\boldsymbol{z}_t$ assuming $\boldsymbol{s}_t^{[m]}$ to get $\Theta^{[m]}$. As a result, we obtain a temporary set of particles for time t,

$$\left\{S_t^{[m]}, \text{where } m \in 1, \ldots, M_t\right\}_{AUX}. \tag{3.21}$$

We conduct landmark updates depending on each different condition as follows.

For the observed landmark, if the probability of the observation given the most likely correspondence $p(\boldsymbol{z}_t \mid \hat{n}_t, \hat{n}_{1:t-1}^{[m]}, \boldsymbol{s}_{0:t}^{[m]}, \boldsymbol{z}_{0:t-1}, \boldsymbol{u}_{1:t})$ is below threshold, we add it to the map as a new landmark. We initialize the new landmark as

$$\boldsymbol{\mu}_{n,t}^{[m]} = \boldsymbol{g}^{-1}(\boldsymbol{z}_t, \boldsymbol{s}_t^{[m]}), \tag{3.22}$$

$$\Sigma_{n,t}^{[m]} = (\mathtt{G}_t^{[m]})^{-1} \mathtt{R}_t ((\mathtt{G}_t^{[m]})^{-1})^T. \tag{3.23}$$

If the probability is above threshold, we combine the new observation with the old landmark position estimate using the extended Kalman filter update equation (see section A.5 for a derivation):

$$\mathtt{K}_t^{[m]} = \Sigma_{n_t,t-1}^{[m]} \mathtt{G}_t^{[m]T} (\mathtt{Q}_t^{[m]})^{-1}, \tag{3.24}$$

$$\boldsymbol{\mu}_{n_t,t}^{[m]} = \boldsymbol{\mu}_{n_t,t-1}^{[m]} + \mathtt{K}_t^{[m]} (\boldsymbol{z}_t - \hat{\boldsymbol{z}}_t^{[m]}), \tag{3.25}$$

$$\Sigma_{n_t,t}^{[m]} = (\mathtt{I} - \mathtt{K}_t^{[m]} \mathtt{G}_t^{[m]}) \Sigma_{n_t,t-1}^{[m]}. \tag{3.26}$$

For the landmarks that are not observed at current time t, their estimates are not affected, so we inherit the estimates at the previous time step as

$$p(\boldsymbol{\theta}_n \mid \boldsymbol{s}_{0:t}, \boldsymbol{z}_{0:t}, n_{1:t}) = p(\boldsymbol{\theta}_n \mid \boldsymbol{s}_{0:t-1}, \boldsymbol{z}_{0:t-1}, n_{1:t-1}). \tag{3.27}$$

Since the landmarks are represented by Gaussian distributions, this trivial update implies

$$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]}, \tag{3.28}$$

$$\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]}. \tag{3.29}$$

Resample based on importance factor

For each temporary particle m, we sampled a new pose from the proposal distribution $p(s_t \mid s_{t-1}^{[m]}, u_t)$ as (3.12), which does not take into account the observation z_t. Therefore, the temporary particle set (3.21) does not exactly distribute as our target distribution $p(s_{0:t}^{[m]} \mid z_{0:t}, u_{1:t}, n_{1:t})$. We calculate the factor (generally referred to as the *importance weight*) that compensates this gap. Assuming that particles in S_{t-1} is distributed as the path posterior at time $t-1$, $p(s_{0:t-1}^{[m]} \mid z_{0:t-1}, u_{1:t-1}, n_{1:t-1})$, after sampling from the proposal distribution, our current path posterior at time t, $p(s_{0:t}^{[m]} \mid z_{0:t-1}, u_{1:t}, n_{1:t-1})$ is distributed as

$$p(s_{0:t}^{[m]} \mid z_{0:t-1}, u_{1:t}, n_{1:t-1}) = p(s_t^{[m]} \mid s_{t-1}^{[m]}, u_t)\, p(s_{0:t-1}^{[m]} \mid z_{0:t-1}, u_{1:t-1}, n_{1:t-1}). \tag{3.30}$$

So we compute the importance weight $w_t^{[m]}$ as

$$\begin{aligned} w_t^{[m]} &= \frac{\text{target distribution}}{\text{proposal distribution}} \\ &= \frac{p(s_{0:t}^{[m]} \mid z_{0:t}, u_{1:t}, n_{1:t})}{p(s_{0:t}^{[m]} \mid z_{0:t-1}, u_{1:t}, n_{1:t-1})} \\ &= \frac{\eta\, p(z_t \mid s_{0:t}^{[m]}, z_{0:t-1}, u_{1:t}, n_{1:t})\, p(s_{0:t}^{[m]} \mid z_{0:t-1}, u_{1:t}, n_{1:t})}{p(s_{0:t}^{[m]} \mid z_{0:t-1}, u_{1:t}, n_{1:t-1})} \\ &= \eta\, p(z_t \mid s_{0:t}^{[m]}, z_{0:t-1}, n_{1:t}). \end{aligned} \tag{3.31}$$

Since (3.31) is equivalent to (3.17), we finally obtain

$$w_t^{[m]} \propto \left|2\pi \mathbb{Q}_t^{[m]}\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(z_t - \hat{z}_t^{[m]}\right)^T \left(\mathbb{Q}_t^{[m]}\right)^{-1} \left(z_t - \hat{z}_t^{[m]}\right)\right\}. \tag{3.32}$$

The importance weights are normalized to sum to 1, then we sample M_t particles, with replacement, from the temporary particle set according to the normalized weights. The result is a new set of particles $\{S_t^{[m]}, m \in 1, \ldots, M_t\}$ that represents the posterior (3.1) at time t.

3.3 Shi-Tomasi Point Detector

Shi-Tomasi features [9] are 2D point features that are used in the tracking algorithm originally developed by Lucas, Kanade and Tomasi [27, 28]. The tracking algorithm uses a simple criterion to choose point features that, for affine motion, are optimal. Under the affine image motion model, a point $x = [x, y]^T$ in image I is expected to be

found at point $\mathtt{A}\boldsymbol{x}+\boldsymbol{d}$ in the next image J, where the 2×2 matrix $\mathtt{A} = \mathtt{D}+\mathtt{I}$ reflects the deformation of the original point $\boldsymbol{x}$, and $\boldsymbol{d} = [d_x, d_y]^T$ is the translation of the feature window's center. The 2×2 matrix

$$\mathtt{D} = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix} \tag{3.33}$$

is a matrix which represents the degree of deformation. If D is a zero matrix, A becomes the identity matrix indicating that the image motion is a pure translation by $\boldsymbol{d}$. Tracking is defined as minimizing the *dissimilarity*

$$\epsilon = \iint_W \left(J(\mathtt{A}\boldsymbol{x}+\boldsymbol{d}) - I(\boldsymbol{x})\right)^2 w(\boldsymbol{x})d\boldsymbol{x}, \tag{3.34}$$

where W is the given feature window and $w(\boldsymbol{x})$ is a weighting function to emphasize a particular region of the window. Tracking means finding the affine parameters A and $\boldsymbol{d}$ which minimize the dissimilarity ϵ.

After linearizing $J(\mathtt{A}\boldsymbol{x}+\boldsymbol{d})$ with respect to D and $\boldsymbol{d}$, and setting the result to zero, we could obtain six linear equations in the six unknown elements of D and $\boldsymbol{d}$. However, for small windows and small distortion, we can assume $\mathtt{D} = 0_{2\times 2}$, so the problem reduces to calculation of only the translation vector $\boldsymbol{d}$. Calculating the derivative of ϵ with respect to $\boldsymbol{d}$ using the first order approximation of $J(\boldsymbol{x}+\boldsymbol{d})$ around $\boldsymbol{x}$ yields

$$\begin{aligned}
\frac{\partial \epsilon}{\partial \boldsymbol{d}} &= 2\iint_W \left(J(\boldsymbol{x}+\boldsymbol{d}) - I(\boldsymbol{x})\right) \frac{\partial J}{\partial \boldsymbol{d}}(\boldsymbol{x}+\boldsymbol{d})\, w(\boldsymbol{x})d\boldsymbol{x} \\
&\approx 2\iint_W \left(J(\boldsymbol{x}) + \frac{\partial J(\boldsymbol{x})}{\partial x}d_x + \frac{\partial J(\boldsymbol{x})}{\partial y}d_y - I(\boldsymbol{x})\right) \\
&\quad \frac{\partial}{\partial \boldsymbol{d}}\left(J(\boldsymbol{x}) + \frac{\partial J(\boldsymbol{x})}{\partial x}d_x + \frac{\partial J(\boldsymbol{x})}{\partial y}d_y\right) w(\boldsymbol{x})d\boldsymbol{x} \\
&= 2\iint_W \left(J(\boldsymbol{x}) - I(\boldsymbol{x}) + \begin{bmatrix} g_x & g_y \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix}\right) \begin{bmatrix} g_x \\ g_y \end{bmatrix} w(\boldsymbol{x})d\boldsymbol{x},
\end{aligned} \tag{3.35}$$

where g_x and g_y are the components of the gradient $\frac{\partial J}{\partial \boldsymbol{x}}$ evaluated at $\boldsymbol{x}$. Letting the derivative be equal to zero, we obtain a linear system in the two unknown elements of $\boldsymbol{d}$ as

$$\mathtt{T}\boldsymbol{d} = \boldsymbol{a}, \tag{3.36}$$

where

$$\mathtt{T} = \iint_W \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} w(\boldsymbol{x})d\boldsymbol{x}, \tag{3.37}$$

and

$$\boldsymbol{a} = \iint_W \left(I(\boldsymbol{x}) - J(\boldsymbol{x})\right) \begin{bmatrix} g_x \\ g_y \end{bmatrix} w(\boldsymbol{x})d\boldsymbol{x}. \tag{3.38}$$

Shi and Tomasi write that good features are features that make (3.36) reliably solved, hence the integrated local gradient matrix (3.37) should be well-conditioned. However, selecting a feature point based on the condition number of the matrix T is not adequate since unstable points that have a low contrast in two orthogonal directions can also be related to a matrix with a good condition number. So the stability of point features and the solvability of the system (3.36) should be related to the local gradient structure of the feature window as follows:

- Both eigenvalues of T in (3.37) should be large, because the variation of the intensity across the window should be well above the image noise level.
- The eigenvalues of T should not differ by large orders of magnitude, because a feature point should have comparably strong intensity variations in both orthogonal directions across it.

Practically, intensity variation cannot become arbitrarily large because of the finite magnitude limit of image intensity. So one strategy is to check if the smaller eigenvalue is above a certain threshold. Let the two eigenvalues of T in (3.37) be λ_1 and λ_2 for some point $\boldsymbol{x}$ in an image. We take the point as a feature point if

$$\min(\lambda_1, \lambda_2) > \lambda, \tag{3.39}$$

where λ is a predefined threshold. Shi-Tomasi feature are points selected in an image according to the criterion (3.39). They are points with strong vertical and horizontal intensity variation such as corners or salt-and-pepper textured regions.

Shi-Tomasi point features were introduced in the context of detecting and tracking points with well-defined texture. But the point feature detection technique itself is also useful for other purposes. The strength of the Shi-Tomasi point detector is its simplicity and light computational cost. It is simple because there are only two parameters to be specified: the feature window size and the threshold λ. This simplicity makes it easy to optimize the detector. The cost of computation is very low because we simply need to calculate the integral of the local gradient matrix over a small window. Other feature detectors such as SIFT are more computationally expensive. SIFT requires construction of a scale space representation of each image, multiple convolutions, and extraction of a rich descriptor of the local image statistics around each point of interest.

We use the Shi-Tomasi point detector to detect salient features (corners and salt-and-pepper textured regions) in images captured by trinocular cameras.

3.4 3D Point Vision Sensor

3.4.1 2D Point Matching Based on Epipolar Constraints

At each time t, we capture three images (see Fig. 3.2 for examples) from a trinocular camera rig and remove image distortion from each image using predetermined rectification tables obtained during calibration. We denote the resulting images as

$$(I_{ref}, I_{hor}, I_{ver})_t, \tag{3.40}$$

where I_{ref}, I_{hor} and I_{ver} represent the rectified images originally captured by the reference, horizontal and vertical camera of the trinocular rig, respectively. We then extract a set of Shi-Tomasi points from each image:

$$\begin{aligned} P_{ref} &= \left\{\boldsymbol{p}_{ref,i}, \text{where each index } i \in 1, \ldots, N_{ref}\right\}, \\ P_{hor} &= \left\{\boldsymbol{p}_{hor,j}, \text{where each index } j \in 1, \ldots, N_{hor}\right\}, \\ P_{ver} &= \left\{\boldsymbol{p}_{ver,k}, \text{where each index } k \in 1, \ldots, N_{ver}\right\}. \end{aligned}$$

(a) (b) (c)

Figure 3.2: Original trinocular image set captured in the lab. (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

N_{ref}, N_{hor} and N_{ver} are the number of Shi-Tomasi points extracted from each reference, horizontal and vertical image, respectively.

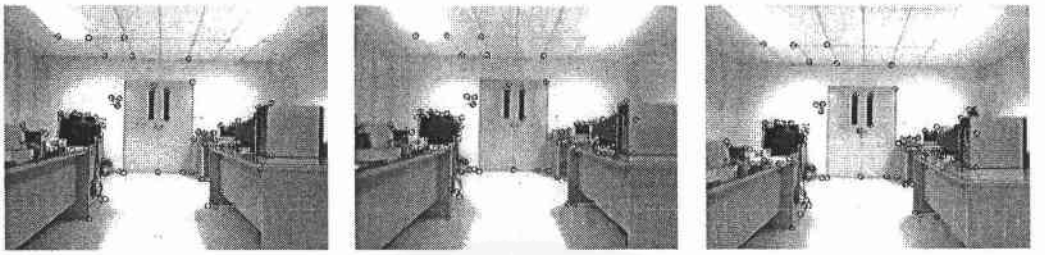
Now we seek three-way correspondences among the Shi-Tomasi points found in each image. Given a feature point $\boldsymbol{p}_{ref}$ in I_{ref}, a possible three-way correspondence $(\boldsymbol{p}_{ref}, \hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})$ is obtained as follows. We draw an epipolar line corresponding to the point $\boldsymbol{p}_{ref}$ in I_{hor} and choose a set of points $\hat{P}_{hor}$ which are close enough to the epipolar line. The point set $\hat{P}_{hor}$ in the horizontal image represents a set of putative matches in the horizontal image for the point $\boldsymbol{p}_{ref}$. We do the same thing for the vertical image as well to get points $\hat{P}_{ver}$, a set of putative matches in the vertical image for the point $\boldsymbol{p}_{ref}$.

For each point in $\hat{P}_{hor}$, we then draw an epipolar line on the vertical image and search for points in the point set $\hat{P}_{ver}$ which are close enough to the epipolar line to get $(\boldsymbol{p}_{ref}, \{(\hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})\})$, a set of horizontal-vertical match point pairs for each reference point $\boldsymbol{p}_{ref} \in P_{ref}$. We then filter this set using an image similarity criterion based on normalized cross correlation between the 11×11 windows around the feature points. To get a unique three-way correspondence for each point $\boldsymbol{p}_{ref}$, we discard the points $\boldsymbol{p}_{ref}$ where the number of elements (possible correspondence pairs) in the filtered correspondence set $\{(\hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})\}$ is not one. Let us write the reference points that survive this process as $\hat{P}_{ref}$. We then finally get a unique three-way correspondence $(\hat{\boldsymbol{p}}_{ref}, \hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})_i$ for each point $\hat{\boldsymbol{p}}_{ref} \in \hat{P}_{ref}$, which constitutes our desired set of correspondences $\{(\hat{\boldsymbol{p}}_{ref}, \hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})_i\}_t$, where $i \in 1, \ldots, N_t^c$. N_t^c is the total number of three-way correspondences obtained.

Typically we begin with about 200 Shi-Tomasi points in each of the images in a trinocular set and end up with 60–150 three-way correspondences depending on the texturedness of the scene. Typically, visual inspection reveals that our method yields less than 10% false correspondences. See Fig. 3.3 for an example.

3.4.2 3D Point Reconstruction from 2D Image Points

We reconstruct 3D landmark points based on the triplets $\{(\hat{\boldsymbol{p}}_{ref}, \hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})_i\}_t$, where $i \in 1, \ldots, N_t^c$, obtained in the previous step. For each trinocular point triplet $(\hat{\boldsymbol{p}}_{ref}, \hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})_i$, a 3D landmark point $\boldsymbol{z}_{t,i}$ in robot-relative coordinates is recon-



(a) (b) (c)

Figure 3.3: Rectified trinocular image set with Shi-Tomasi points whose three-way correspondences are established (marked by circles). (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

structed as follows. We triangulate points $\hat{\boldsymbol{p}}_{ref}$ and $\hat{\boldsymbol{p}}_{hor}$ to get an initial estimate of the 3D point $\boldsymbol{z}_i$, then we use Levenburg-Marquardt nonlinear least squares optimization to refine the estimate of $\boldsymbol{z}_i$ to maximize the likelihood of the 2D observations $(\hat{\boldsymbol{p}}_{ref}, \hat{\boldsymbol{p}}_{hor}, \hat{\boldsymbol{p}}_{ver})_i$ assuming spherical Gaussian error in the measured image coordinates. We also obtain an estimate of confidence in $\boldsymbol{z}_{t,i}$ by propagating the assumed measurement error through the maximum likelihood estimation procedure using the standard first-order approximation [29]. As a result, the distribution of each 3D landmark point is approximated to first order by a 3D Gaussian distribution with mean $\boldsymbol{z}_{t,i}$ and 3×3 covariance matrix $\mathrm{R}_{t,i}$.

3.4.3 FastSLAM with 3D Landmark Observations

In FastSLAM, the sensor model is fully described by the conditional probability $p(\boldsymbol{z}_{t,1}, \ldots, \boldsymbol{z}_{t,N_t^c} \mid \boldsymbol{s}_t, \Theta_{t-1}, \boldsymbol{n}_t)$, explicitly conditioning on $\boldsymbol{n}_t$, the N_t^c-element vector of correspondences between observations $\boldsymbol{z}_{t,i}$ and landmarks stored in Θ_{t-1}. We assume that each observation is conditionally independent, which allows us to write the sensor model as

$$p(\boldsymbol{z}_{t,1}, \ldots, \boldsymbol{z}_{t,N_t^c} \mid \boldsymbol{s}_t, \Theta_{t-1}, \boldsymbol{n}_t) = \prod_{i=1}^{N_t^c} p(\boldsymbol{z}_{t,i} \mid \boldsymbol{s}_t, \Theta_{t-1}, \boldsymbol{n}_t). \tag{3.41}$$

Let $\boldsymbol{\theta}_n$ be the random variable representing the world coordinate position of the nth landmark, our current estimate of which is represented by a Gaussian distribution with mean $\boldsymbol{\mu}_{n,t-1}$ and covariance $\Sigma_{n,t-1}$, and let N be the total number of landmarks in the map. Then we denote the stochastic map at time $t-1$ as

$$\Theta_{t-1} = \left\{ (\boldsymbol{\mu}_{n,t-1}, \Sigma_{n,t-1}), \text{where each index } n \in 1, \ldots, N \right\}. \tag{3.42}$$

The sensor model is assumed to be a deterministic measurement function $\boldsymbol{g}(\boldsymbol{\theta}_n, \boldsymbol{s}_t)$ corrupted by Gaussian noise $\boldsymbol{\epsilon}_t$ with mean zero and covariance $\mathrm{R}_{t,i}$:

$$\boldsymbol{z}_{t,i} = \boldsymbol{g}(\boldsymbol{\theta}_{n_{t,i}}, \boldsymbol{s}_t) + \boldsymbol{\epsilon}_{t,i}, \text{where } \boldsymbol{\epsilon}_{t,i} \sim \mathcal{N}(\boldsymbol{\epsilon}_{t,i}; \mathbf{0}, \mathrm{R}_{t,i}). \tag{3.43}$$

$\boldsymbol{\theta}_{n_{t,i}}$ is the random variable representing the position of the specific landmark associated with element i of the current observation. The measurement function $\boldsymbol{g}(\boldsymbol{\theta}_{n_{t,i}}, \boldsymbol{s}_t)$ is

nonlinear for most cases of interest including our six degree of freedom trinocular camera rig. To make the formulation of $p(\boldsymbol{z}_{t,i} \mid \boldsymbol{s}_t, \Theta_{t-1}, \boldsymbol{n}_t)$ tractable, we linearize the function $\boldsymbol{g}$ with respect to the landmark $\boldsymbol{\theta}_{n_{t,i}}$. Linearizing $\boldsymbol{g}$ around $\boldsymbol{\theta}_{n_{t,i}} = \boldsymbol{\mu}_{n_{t,i},t-1}$ yields

$$\begin{aligned} \boldsymbol{g}(\boldsymbol{\theta}_{n_{t,i}}, \boldsymbol{s}_t) &\approx \boldsymbol{g}(\boldsymbol{\mu}_{n_{t,i},t-1}, \boldsymbol{s}_t) + \boldsymbol{g}'(\boldsymbol{\mu}_{n_{t,i},t-1}, \boldsymbol{s}_t)(\boldsymbol{\theta}_{n_{t,i}} - \boldsymbol{\mu}_{n_{t,i},t-1}) \\ &= \hat{\boldsymbol{z}}_{t,i} + \mathtt{G}_{t,i}(\boldsymbol{\theta}_{n_{t,i}} - \boldsymbol{\mu}_{n_{t,i},t-1}), \end{aligned}$$

where $\hat{\boldsymbol{z}}_{t,i} = \boldsymbol{g}(\boldsymbol{\mu}_{n_{t,i},t-1}, \boldsymbol{s}_t)$ is $\boldsymbol{g}$ evaluated at $\boldsymbol{\theta}_{n_{t,i}} = \boldsymbol{\mu}_{n_{t,i},t-1}$, and $\mathtt{G}_{t,i} = \boldsymbol{g}'(\boldsymbol{\mu}_{n_{t,i},t-1}, \boldsymbol{s}_t)$ is the Jacobian of $\boldsymbol{g}$ evaluated at the same point. Since the landmark $\boldsymbol{\theta}_{n_{t,i}}$ is assumed to conform to the normal distribution $\mathcal{N}(\boldsymbol{\theta}_{n_{t,i}}; \boldsymbol{\mu}_{n_{t,i},t-1}, \Sigma_{n_{t,i},t-1})$, it follows that $\boldsymbol{g}(\boldsymbol{\theta}_{n_{t,i}}, \boldsymbol{s}_t)$ after the linearization also follows the normal distribution

$$\boldsymbol{g}(\boldsymbol{\theta}_{n_{t,i}}, \boldsymbol{s}_t) \sim \mathcal{N}(\boldsymbol{g}; \hat{\boldsymbol{z}}_{t,i}, \mathtt{G}_{t,i}\Sigma_{n_{t,i},t-1}\mathtt{G}_{t,i}^{-1}).$$

Therefore the conditional probability $p(\boldsymbol{z}_{t,i} \mid \boldsymbol{s}_t, \Theta_{t-1}, \boldsymbol{n}_t)$ is now approximated to be a normal distribution with mean $\hat{\boldsymbol{z}}_{t,i}$ and covariance $\mathtt{G}_{t,i}\Sigma_{n_{t,i},t-1}\mathtt{G}_{t,i}^{-1} + \mathtt{R}_{t,i}$ as

$$p(\boldsymbol{z}_{t,i} \mid \boldsymbol{s}_t, \Theta_{t-1}, \boldsymbol{n}_t) \sim \mathcal{N}(\boldsymbol{z}_{t,i}; \hat{\boldsymbol{z}}_{t,i}, \mathtt{G}_{t,i}\Sigma_{n_{t,i},t-1}\mathtt{G}_{t,i}^{-1} + \mathtt{R}_{t,i}). \tag{3.44}$$

In our case, the observation at time t is a set of 3D landmark points in robot coordinates whose measurement function is explicitly written for each landmark as

$$\boldsymbol{g}(\boldsymbol{\theta}_{n_{t,i}}, \boldsymbol{s}_t) = \mathtt{A}_{\phi,t}(\boldsymbol{\theta}_{n_{t,i}} - \boldsymbol{s}_{x,t}), \tag{3.45}$$

where $\boldsymbol{s}_{x,t} = [s_{x,t}, s_{y,t}, s_{z,t}]^T$ is the translation components of the robot pose $\boldsymbol{s}_t$ in the world coordinates, and $\mathtt{A}_{\phi,t}$ is the rotation matrix that maps a translated 3D landmark to the robot-relative coordinates based on the orientation of the robot. Given the stochastic map Θ_{t-1}, the landmark data association $\boldsymbol{n}_t$, the measurement equation (3.45), and the new landmark covariances $\mathtt{R}_{t,i}, i \in 1, \ldots, N_t^c$ estimated by triangulation with the trinocular stereo vision rig, the sensor model is fully described according to (3.41) and (3.44).

After 2D feature detection, correspondence estimation, and triangulation, we obtain a set of 3D point landmark observations with associated error covariance matrices. The set of landmarks with covariances $\boldsymbol{z}_{t,i}, \mathtt{R}_{t,i}, i \in 1, \ldots, N_t^c$ constitute the robot's observation at time t, which is input to FastSLAM. From this point on, our system is identical to Thrun et al.'s FastSLAM 1.0 algorithm [1].