École Doctorale Systèmes

Thèse

pour obtenir le grade de

Docteur de l'Institut National Politechnique de Toulouse

Spécialité: Systèmes Automatiques

présentée et soutenue publiquement le 2 février 2007

Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach

Joan Solà Ortega

Préparée au Laboratoire d'Analyse et d'Architecture de Systèmes du CNRS
sous la direction de M. André Monin et M. Michel Devy

Jury

M. Jean-Yves Fourniols	Président
M. Andrew Davison	Rapporteur
M. Patrick Rives	Rapporteur
M. Raja Chatila	Examinateur
M. Gérard Favier	Examinateur
M. Alberto Sanfeliu	Examinateur
M. André Monin	Directeur de Thèse
M. Michel Devy	Codirecteur de Thèse

Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach.

Joan Solà

February 2, 2007

Contents

List of Symbols

Sets, Algebra and Geometry

$\mathbb{R}, \mathbb{C}, \mathbb{H}$	The sets of Real and Complex numbers. The set of quaternions.
$\mathbb{R}^n$	The space of all n-tuples of Real numbers.
$\mathbb{E}^n$	The n-dimensional Euclidean space.
$\mathbb{P}^n$	The n-dimensional Projective space.
p, v	A generic point; a generic vector.
$\mathbf{p}$, $\mathbf{v}$	The coordinates $\mathbf{p}, \mathbf{v} \in \mathbb{R}^n$ of p; of v.
$\mathbf{x}$	A vector.
$\mathbf{X}$	A matrix.
$\mathbf{x}^\top$, $\mathbf{X}^\top$	The transpose of $\mathbf{x}$; of $\mathbf{X}$.
X	A stacked vector, *e.g.* $X^\top = [\mathbf{p}_1^\top \ \mathbf{p}_2^\top]$.
$\mathbf{q}, \mathbf{e}, \mathbf{R}$	The quaternion, Euler angles and Rotation matrix.
$\mathbf{q}^*$	The conjugate of the quaternion $\mathbf{q}$.
$\mathbf{q}_1 \cdot \mathbf{q}_2$	The product of quaternions in the quaternion space algebra.
$\mathcal{F}$	A reference frame.
$\mathcal{F}\{FLU\}$	An oriented reference frame (*e.g.* X-Front, Y-Left, Z-Up).
$\mathcal{F}$	The state vector of a reference frame, *e.g.* $\mathcal{F}^\top = [\mathbf{x}^\top \ \mathbf{q}^\top]$.
$\mathcal{F}(t)$	A mobile reference frame.
$\mathbf{p}^{\mathcal{F}}$	The coordinates of p expressed in the $\mathcal{F}$ frame.
$\mathcal{F}^{\mathcal{G}}$	The $\mathcal{F}$ frame expressed with respect to the $\mathcal{G}$ frame.
$\mathbf{t}, \mathbf{R}$	The translation vector and rotation matrix.
$\mathbf{H}$	An homogeneous matrix.
$\mathbf{H}^{\mathcal{F}\mathcal{G}}$	The homogeneous matrix from frame $\mathcal{G}$ to frame $\mathcal{F}$.
$\underline{\mathbf{p}}, \underline{\mathbf{v}}$	The homogeneous coordinates $\underline{\mathbf{p}}, \underline{\mathbf{v}} \in \mathbb{P}^3$ of p and v.
$f(\cdot)$	A scalar function.
$\mathbf{f}(\cdot)$	A vector function.
$\mathbf{f}(\mathbf{x}, \mathbf{y})$	A multivariate vector function.
$\dot{\mathbf{f}}$	Time-derivative of $\mathbf{f}$.
$\mathbf{F}$	The Jacobian matrix of $\mathbf{f}$.
$\mathbf{F}_\mathbf{x}$	The Jacobian matrix of $\mathbf{f}$ with respect to $\mathbf{x}$.
$\mathbf{x}^+$	The updated value of $\mathbf{x}$, *e.g.* $\mathbf{x}^+ = \mathbf{f}(\mathbf{x})$.
α, β, τ	Scalar tuning parameters.
$\mathbf{c}, \mathbf{d}, \mathbf{k}$	Vector calibration parameters.
ϕ, θ, ψ	Angles.
ω	Angular rate.

Vision

X, Y, Z	3D coordinates in the camera's sensor frame of a point in space.
$X^{\mathcal{F}}, Y^{\mathcal{F}}, Z^{\mathcal{F}}$	3D coordinates in $\mathcal{F}$ frame of a point in space.
x, y	2D normalized image coordinates.
x_d, y_d	Distorted coordinates in the normalized image plane.
u, v	Pixellic image coordinates.
$\mathcal{I}\{RD\}$	The oriented image frame: u-Right; v-Down.
$\mathbf{I}$	The image's photometrical properties.
$\mathbf{I}(x, y)$	The image in metric coordinates.
$\mathbf{I}(u, v)$	The image in pixel coordinates.
$\alpha_u, \alpha_v, \alpha_\theta, u_0, v_0$	Camera intrinsic calibration parameters.
$d_2, d_4, \cdots$	Camera distortion parameters.
$c_2, c_4, \cdots$	Camera correction parameters.
$\mathbf{k}, \mathbf{d}, \mathbf{c}$	Camera Intrinsic, Distortion and Correction parameter vectors.
$\mathbf{K}$	Intrinsic matrix.
$\mathbf{P}_0$	Normalized projection matrix.
$d(\cdot), c(\cdot)$	Distortion and correction functions.

Random variables and estimation

pdf	Abbr. of 'Probability Density Function'.
$p_X(\mathbf{x})$	*pdf* for the random variable X as a function of the possible values $\mathbf{x}$.
$p(\mathbf{x})$	Shortcut for the term above.
$\mathbf{x}$	Alternate name for the random variable X.
$\bar{\mathbf{x}}$	The mean or expected value of $\mathbf{x}$
$\mathbf{X}$	The covariances matrix of $\mathbf{x}$.
$\hat{\mathbf{x}}$	An estimate of the random variable $\mathbf{x}$.
$\mathcal{N}$	Identifier for Normal or Gaussian *pdf.*
$\mathcal{N}(\mathbf{x} - \bar{\mathbf{x}}, \mathbf{X})$	Gaussian *pdf* as a function of variable $\mathbf{x}$ and parameters $\bar{\mathbf{x}}$ and $\mathbf{X}$.
$\mathbf{x} \sim \mathcal{N}\{\bar{\mathbf{x}}; \mathbf{X}\}$	Shortcut for '$\mathbf{x}$ *is Gaussian with mean* $\bar{\mathbf{x}}$ *and covariances matrix* $\mathbf{X}$'.
$\epsilon, \upsilon, \omega$	Random perturbations or noises.
σ_x	Standard deviation of the scalar variable x.
$n\sigma$	Iso-probable ellipse or ellipsoid of Mahalanobis distance equal to n.

Localization and Mapping

$\mathcal{W}, \mathcal{R}, \mathcal{C}, \mathcal{S}, \mathcal{O}$	The world-, robot-, camera-, sensor- and object- reference frames.
$\mathbf{x}, \mathbf{p}, \mathbf{r}$	The robot-, landmark- and object- positions.
$\mathbf{q}, \mathbf{e}, \mathbf{R}$	Quaternion, Euler angles and Rotation matrix for robot's orientation.
$\mathbf{x}_R, \mathbf{q}_R$	The Right-hand camera reference frame's position and orientation.
$\mathcal{R}, \mathcal{R}(t)$	State vector of the robot frame, *i.e.* $\mathcal{R}^\top = [\mathbf{x}^\top \ \mathbf{q}^\top]$.
$\mathcal{M}$	State vector of the set of landmark positions, *i.e.* $\mathcal{M}^\top = [\mathbf{p}_1^\top \cdots \mathbf{p}_n^\top]$.
X	The SLAM map, *e.g.* $X^\top = [\mathcal{R}^\top \ \mathcal{M}^\top]$ or $X^\top = [\mathcal{R}^\top \ \mathbf{q}_R^\top \ \mathcal{M}^\top]$.
$X \sim \mathcal{N}\{\hat{X}; \mathbf{P}\}$	The EKF-SLAM map.
$\mathbf{u} \sim \mathcal{N}\{\bar{\mathbf{u}}; \mathbf{U}\}$	The robot's motion odometry, with its mean and covariances matrix.

$\mathbf{f}(\cdot)$	A generic time-evolution differences function.
$\mathbf{g}(\cdot)$	A generic inverse observation function.
$\mathbf{h}(\cdot)$	A generic observation function.
$\mathbf{j}(\cdot)$	A generic frame transformation function.
$\mathbf{F}, \mathbf{G}, \mathbf{H}, \mathbf{J}$	The Jacobian matrices of the generic functions above.
$\mathcal{R}^+$	The updated value of $\mathcal{R}$, *e.g.* $\mathcal{R}^+ = \mathbf{f}(\mathcal{R}, \mathbf{u})$.
$\mathbf{y}, \mathbf{R}$	Performed measurement and its noise covariances matrix.
$\mathbf{b} \sim \mathcal{N}\{\mathbf{y}, \mathbf{R}\}$	Noisy measurement as a Gaussian *pdf*.
$\mathbf{e} \sim \mathcal{N}\{\bar{\mathbf{e}}; \mathbf{E}\}$	Measurement expectation.
$\mathbf{z} \sim \mathcal{N}\{\mathbf{0}; \mathbf{Z}\}$	Measurement innovation.
α, β	Aspect ratio and geometric base of the geometric ray.
λ, Λ	Likelihood and weight of the geometric ray terms.
γ, μ, τ	Vanishing and shape factors and pruning threshold for ray weights.
ρ	Federative coefficient for Federated Information Sharing updates.

Moving Objects Tracking

$\mathbf{r}, \mathbf{v}$	The moving object's position and velocity.
$\mathcal{O}, \mathcal{O}(t)$	Moving object's state vector, *i.e.* $\mathcal{O}^\top = [\mathbf{r}^\top \ \mathbf{v}^\top]$.
S_L, S_I, S_E, S_G	Detection Mechanisms binary matrices.
V_I, V_E, V_G	Detection Mechanisms velocity matrices.
F_L, F_O	Existing features matrices for landmarks and objects.
D_L, D_O, V_O	Output binary and velocity matrices for landmarks and objects.

Introduction

This report you are handling resumes three years of work.

In these three years, I tried to reach my objective of providing a man-made machine such as a robot with the means of *understanding complex and dynamic visual scenes*. The fact of a visual scene being *complex* is not much a matter of discussion (but the complexity itself is) if we just take a look around us in any situation of our everyday life. The *dynamism*, which comes from the fact that we are alive and need interaction with other living creatures, can be simply understood as the necessary movement of the different bodies in seek of their vital needs.

If we take a close look to the visual scene that a moving being may perceive, we will soon notice two main sources of motion. On one hand, any three-dimensional motion at the level of the own visual sensors (eyes or cameras) leads to a two-dimensional motion of the whole perceived image, which has a particular coherence. This coherence has been studied by different means in the past centuries, and has been mathematically understood and formalized short ago in what we know today as *Multiple View Geometry* (MVG) [Hartley and Zisserman 2000]: the world's 3D structure, the sensed material's 2D structure and the sensor motion are intimately linked, and MVG is just the mathematical enlightenment of this linkage. On the other hand, new sources of 2D motion are generated by the existence in the field of view of other moving bodies, whose motions are independent from the observer's, and unknown by him. Obtaining the characteristics of the motions of these other bodies is not an easy task if we restrict our reasoning to the two dimensions of the image spaces. We need to understand what is really happening in three dimensions, and a complete reconstruction of the 3D geometry of the scene becomes almost inevitable.

The *problem* to be solved becomes therefore of triple scope: first, we will have to reconstruct the static part of this scene, what we will call the 'world'; second, we will need to know 'where' in this world we are located; and finally, we will have to detect and keep track of those other moving objects, try to identify or at least have some description of them, and reasonable good information on their positions and velocities. And why? Because we need to be able to predict their future positions in order to maintain a safe interaction (if we are so kind) or to produce a certain impact on their behavior (if we have other objectives). In addition, these three tasks have to be solved simultaneously and in real-time, that is to say at the right time the observer is moving and the scene takes place, in order to take the appropriate conclusions and allow us to act, therefore, consequently.

This about the problem itself.

But there are also different *particular ways to look at the problem*. For reasons that often escape to my awareness, but that could easily be imputed to my kind co-director Michel Devy, I found myself imagining our observer in one particular situation: on the motorway, trying to safely drive a vehicle. I soon discovered that there were fundamental consequences of considering such kind of a scenario. Indeed, many solutions to autonomous 3D navigation take what I call the 'hardware-driven' approach, where one sets up the necessary hardware to solve the problem: tachometers, encoders, lasers, radars, inertial units, GPS and so on. In the case of vision and 3D, the natural consequence of this approach is setting up a stereo bench. You get this nice 3D properties, but you have to pay the price for it: two cameras equals double image processing; a fragile structure means endless calibrations; and, above all, its inability to accurately measure large distances leads to its impossibility to consider remote objects. At large distances, the views of both cameras are exactly the same, and a stereo bench provides exactly the same information as one single camera. The question naturally arises: what about using just one camera? As a response, just one evidence: if you ever played video games, you know that the problem of driving in dense traffic is solvable exclusively from monocular vision. No stereo, no GPS, no tachometers. Just monocular vision as shown in the figure. We do not even need to know the metric scale of the scene in order to react properly. This makes the interest for monocular vision arise: can we recover this dynamic 3D world from a single camera in motion? the answer is 'yes', of course, but only when we take what I call the 'intelligence-driven' approach, in which the lack of hardware is substituted by an extra effort of reasoning. Once you have solved the 'intelligence-driven' approach, you can always go back and increase the hardware complexity: you will get all the advantages and none of the drawbacks.

Figure 1: Video-games send us a monocular version of a virtual reality where the scale factor is irrelevant. Our understanding of the scene is out of doubt, and our success in the game is only dependent on our driving skills.

This about my particular way to look at the problem.

About *the way to reason about the problem*, I found we needed to take it also from a triple viewpoint. First, we need to understand the 3D world, their objects and motions, their rules. Second, we need to know how we perceive this world, how vision works, what information we can get from images, what is lost on the way. Third, we need to know how we can make a machine understand anything at all. Let us talk about this.

The 3D world. We are so used to it that we find it is like this because it has to be like this. Even without Isaac Newton, everybody knows that apples fall from trees if we don't collect them before. Beware: we do not need to go into the question of 'why' the world we know is precisely like this (why it is three- and not four-dimensional, for instance[1,2]). But we do need to go into the question of 'how' it actually is, and how it actually works. We need to know how to describe their places, shapes, distances and directions. We need to understand how solids behave, how they move. How they move when we apply an action on them, for instance. We need to get into Newton's work. And Hamilton's. And Descartes'. And Euler's. And so many others. I will talk about this in Chapter 1.

If a world description like the one we try to get above can mechanically explain how a whole set of bodies *move* or *are situated* ones with respect to others, perception, and vision in particular, arises the idea of a *living thing*. If something *sees* in any sense is because it is *useful* for it, because it has to fulfill some objective, because there is a purpose for it: because it is *alive*. Stones do not need to see. They never die anyway! Artificial vision like photography is for me the result of some curious minds being asked themselves 'what' is this amazing little 'capsule' that make us see, 'how' it works, and 'why' we can not pass without it. An even more mysterious issue is 'why' we feel so strongly that we should make one ourselves. Something that astonishes me about vision is the ability to fit the immense world inside this small capsule so that we can know about our exterior. Is that the reduction from 3D to 2D named 'projection' what makes it possible? Perhaps. But what? and why? If you think deeply about that you will get kind of blocked with strange questions. If you don't block, please let me know. I will talk about 'how' we can make artificial eyes work in Chapter 2.

When I first began to think about all these issues, I already had some background on filtering. This is thanks to my director André Monin, who gave me the key to this so often misunderstood probabilistic world. From filtering, I knew that the fact of being able to make predictions provides us with an incredible amount of understanding possibilities as, as Heraclitus said:[3]

"Everything flows, nothing stands still"

which for me, when I first got to know about the idea, stated that things you may be aware of did not come up from nothing but 'became' from a continuous sequence of states. This continuity of change or 'flow', when understood in some way, when reduced to any sort of rule, allows us to make predictions about the near future. Perception is there only to verify whether our predictions were right or not, refining and modifying the inferred rules accordingly. This is *common sense* itself, and I will talk about it in Chapter 3. Let me say that, for us, common sense is just a kind of unconscious reasoning that helps us resolving a vast amalgam of everyday situations. For a machine, however, this unconsciousness is unacceptable: honestly, who would trust a machine that reasons on its own, without being told how to reason 'properly'? Hence in order to tell them 'that', we need to bring this unconscious reasoning into light, understand how it works, and transfer its mechanisms into the machine. This is what filtering is about —from this point of view and if you let me say that.

This about the way to reason about the problem.

So we have a problem and a particular way to look at it, and we have a way to reason about it. Let us call it a *philosophy*. Well, if we want to make something about that, if we want to

[1] It seems that generalized gravitational equations give unstable universes for even dimensions, so we would have only 1D, 3D, 5D and so on possible worlds.

[2] And I say this: the world is of the minimum admissible dimension that makes the apparition of life possible. Lower dimensions (1D) make life impossible; higher dimensions make much more complex worlds and hence much more improbable for life to appear.

[3] Quoted by Plato in *Cratylus*, and by Diogenes Laertius in *Lives of the Philosophers*, Book IX, section 8

get the problem solved, if we want maybe just to make one step forward, we need a method. We need *a way to board the problem*. We need tools. We cannot just keep on philosophizing —although I like it so much indeed. I did not know it before, so I spent my first year trying to use filtering to solve the dynamics problem from exclusively visual information (and I got quite a bunch of ideas) but, one day, the 'II SLAM summer school' came to LAAS.[4] I saw so many people asking themselves the same questions as I did! There was Thrun. There was Konolige. There was Davison. And many others. And they just had the exact tools I needed! I swear that was the end of philosophy for me, but that put me into the SLAM track and I began being productive. SLAM: 'Simultaneous Localization And Mapping' [Durrant-Whyte and Bailey 2006]. In a way, that was the first two parts of my problem being solved! And the ideas I had were fitting very well: SLAM solves the geometric problem of 'how the world is' and of 'where I am' based on perception information, and it does it by filtering, so predicting what is going to happen and correcting the constructed models afterwards. But still my problem was far from being solved: I wanted to use vision and they were talking about lasers and sonars. I wanted to do it fully 3D operative, and they were doing flat, 2D maps. I wanted to go far, to travel on the straight line of my motorway and not necessarily coming back to the start, and they were insisting so much on closing loops and mapping lab corridors. And I wanted to detect and track other moving objects, and nobody was talking about that. (Of course I am talking about my impressions during the SLAM school; SLAM was already beyond that as I got to know it later.) Finally, on the last day of the school, Andrew Davison showed up with his amazing real-time demo: monocular vision and 3D in real-time! So there was somebody up there after all! This gave me the certainty that I was on the right way.[5] I will talk about these tools in Chapter 4.

And this about the necessary tools. The rest would have to be worked out.

I divide thus this report into two parts. The first part contains all that I have talked about so far. The second part is everything I worked out on top of that. Here is what I did:

The first step was to take advantage of some ideas I had about regarding vision from a probabilistic viewpoint. The question was how to infer 3D characteristics from 2D images. I had already focused my attention to single points because, in the 3D world, these are the only 'objects' that can be considered invariant to scale changes and rotation, that have no shape and no size, that cannot be 'partially' occluded. They just have a position, and either you see them or you do not. This was going to significantly reduce the problem complexity but, at the same time, was going to close some doors. I accepted the price and formulated to myself the following question: "Is my problem solvable from exclusively geometric considerations about punctual positions?" From all the work I have done since then, I can say the answer is 'yes, with remarks'. The 'yes', you have one hundred and eighty pages to discover it, but it started with the mental visualization of the probability density functions derived from the observations: the knowledge we can infer from the images. The way Geometry, Vision and Probabilities were setting up a comprehensible framework for the problem is sketched in Chapter 5. The 'remarks', let me keep on being positive by now, I reserve them for the conclusions.

The second step was to incorporate SLAM tools. At that time, I was becoming refractory to use heavy material such as the Particle Filter [Doucet et al. 2001] to solve any kind of filtering

[4] Laboratoire d'Analyse et Architecture de Systèmes, Toulouse, France.

[5] However, thirty months after that time, my opinion of SLAM has changed a lot. Now, SLAM is just a black hole: once you fall inside, you cannot escape. Should we start something different?

problem without first analyzing its pertinence (in Catalan[6] we say this is like "killing flies with gunshots") and I started considering Gaussian approaches, in particular the classical SLAM solution [Smith and Cheeseman 1987] based on the Extended Kalman Filter [Breakwell 1967]. Also, the Gaussian Sum Filter [Alspach and Sorenson 1972] constituted a solid candidate to new SLAM solutions by combining both particle and Gaussian characteristics.

In another order of things, the problem I wanted to solve was initially restricted to the static world and the moving observer, that is to say exactly the SLAM problem, with the aid of exclusively monocular vision and odometry, in motorway-like conditions: a vehicle following a straight line and looking forward. In these situations, the camera motion does not provide the ideal conditions for 3D reconstruction: the basic principle for 3D reconstruction is triangulation, where the triangles are defined by two camera poses and the sought point. In our case, with the sought point almost aligned with the successive camera poses, this triangle is extremely bad defined. The monocular SLAM methods available at the time were avoiding such kind of ill-conditioned situations, and were therefore absolutely helpless. The necessity of mapping partially observed points (points for whom we only know the direction, not the distance) became imperative: without them, the most important part of the scene, that is to say the front of our vehicle, was condemned to be neglected. Research on this problematic led to a new formulation of the EKF-based SLAM algorithm where this partially known information was mapped as a special series of Gaussian densities.

The perception issues that arise from the fact of imposing severe real-time constraints also needed special attention. From the very same principles of filtering, however, these constraints naturally suggested to analyze those image regions where we know the information must be (that is, we can predict where we have to observe). This led to the adoption of the so called 'active search' approaches which are also used in other major works on visual SLAM.

All the research devoted to monocular SLAM is collected in Chapter 6. Several experiments show the pertinence of the proposed methods for solving the mono-SLAM problem in these ill-conditioned trajectories.

I have to say that the whole period of work on monocular SLAM was cool: from the SLAM community I could pick up an amazing amount of nice ideas, sometimes powerful intuitions, that could be rapidly incorporated to my thoughts. But I had to keep an eye to the ultimate problem of the moving objects, for which SLAM had no response. The most important issue is observability, something that, for the monocular case, had already been studied in the 'bearings-only tracking' literature [Le Cadre and Jauffret 1997]. Unfortunately, their conclusions are not very promising. I decided then to use more than one camera running the mono-SLAM algorithms to achieve instantaneous triangulation. This would provide sufficient mid-range 3D observability in the front of the vehicle, while keeping the possibility of observing remote landmarks, potentially at infinity, to improve localization. Additionally, I realized that, by doing monocular SLAM twice, once per camera, the redundancy was so important that I could use the extra information for other purposes. As one of the main drawbacks of stereo vision is mechanical fragility, which fatally derives into continuous re-calibration headaches, I decided to use this redundancy to self-calibrate the geometrical parameters of the stereo bench. This work, which did not demand a lot of effort but just putting the pieces together, is presented in Chapter 7.

[6] Catalan is my language. It is also the language of other seven million people, making it the eighth spoken language in Europe. But the lack of institutional support and the tough effort of Spanish and French governments are killing it. See for instance http://www6.gencat.net/llengcat/publicacions/cle/docs/ecle7.pdf.

The last step was to incorporate detection and tracking of moving objects. A more accurate observability evaluation confirmed that, while monocular solutions could effectively be used, they demanded either unacceptable conditions on the observer's trajectory (continuous, random-like changes of direction) or the development of completely new approaches for which real-time implementation was unclear. This suggested me to take advantage of the solutions developed so far: the methods from the previous chapter allowed the robot to have full 3D observations of a relatively large area in front of the robot. This way, by relaxing the initial goal of achieving the whole dynamic reconstruction from monocular vision (and modestly accepting a binocular solution), I was able to unblock the observability problem. And once we know a system is observable, we can start working on it.

The addition of moving objects involves the inclusion of two radically different mechanisms: *detection* and *tracking*. As we cannot afford missing them, objects need to be detected before they can be tracked at all. In Chapter 8, which closes this research, both mechanisms are discussed and presented:

Starting at the easy one, *tracking* only demands the addition of a small set of operations which are already well known, because they are the same that have been used during the whole work. This is basically assigning a filter to each object and using the filtering steps of prediction, active-search-based measurements and correction.

The *detection* problem is the problem of finding new moving objects. It is by far the most delicate because real-time constraints suggest not to make exhaustive image scans. The approach I took consists of anticipating where in the images a new moving object can appear, searching for something relevant there, and assigning it a certain initial estimate based on possible *a-prioris*. With a philosophy akin to that of active feature search, the mechanisms used to anticipate these situations take advantage of the knowledge the robot has on the system, to optimize the chances of success while respecting the severe constraints imposed by the limited computation resources.

This report you are handling resumes three years of work. I am happy to say that I could fulfill my principal objectives. Here you are how I did it:

A (very short) reading guide

As it can be derived from the introduction, the two parts of this work are unequal. The first part contains mainly known material and should be read in one go. My contributions inside, from my point of view, are just the particular way to re-explain it. If I had to justify why I wrote it, I would have to say this: I'm not sure. But it gives to the second part the solidity I like; It permits to justify in the largest possible sense some of the key decisions I take; It allows the students that will hopefully renew my work to arrive sooner and easier to the heart of the question; And, as this is my first and maybe (hopefully not) the last book I write, this is the way I felt I had to do it.

In any case, this part sets important bases that guide the whole work: terminology, notation, definitions, etc. It also contains one 'true' contribution, in the Vision chapter, when it is time to invert the distortion function.

The second part constitutes my research itself. A final third part has been reserved for the appendices.

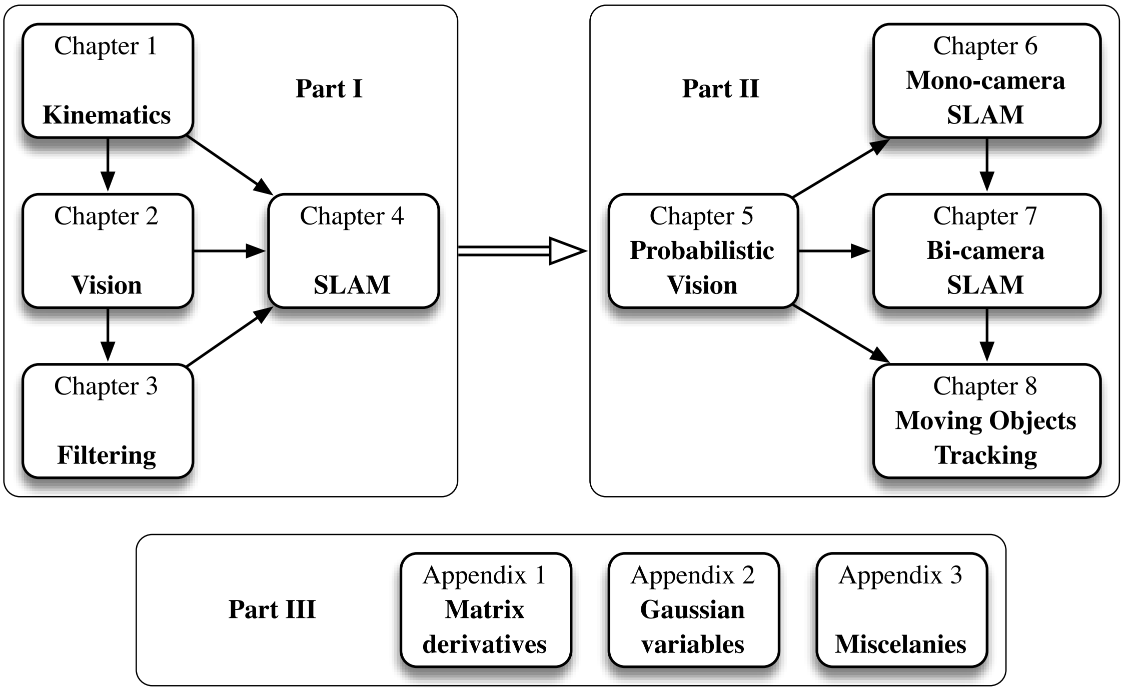

Figure 2: Plan of this work

Part I

Introductory material

Chapter 1

Kinematics

We –or at least *'we, the engineers'*– are more than familiarized with identifying our three-dimensional (3D) world with the mathematical space $\mathbb{R}^3$. However, this identification is weak: while $\mathbb{R}^3$ is just the set of triplets of real numbers, the 3D space includes the notions of *distance* and *direction*, of *size* and *shape*, about which we –now *'we, all proactive creatures living in ecological competition'*– have a deep and native intuition that allows us to understand the world, and hence to live in it. The *3D Euclidean space*, that we will denote $\mathbb{E}^3$, is the mathematical construction that includes these concepts of distance and direction, that resume up to the notion of *metric*.

Engineers are concerned with the invention of tools. Tools that have to be useful in our 3D world, that have to *live* in it. But a tool being a non-living creature, it has by no means these deep and native intuitions, and the engineer has to provide something equivalent. This is what $\mathbb{E}^3$ is about: it is the world were engineers play, the place where their tools are ultimately conceived.

1.1 Introduction. The three-dimensional Euclidean space

Let me introduce you to a formal description of this Euclidean world. The goal is simply to enrich $\mathbb{R}^3$ with a metric. And in order to do so, some preliminary concepts need to be introduced.

In mathematics, the *Cartesian coordinate system* or *frame* can be used to uniquely determine each point in the space through three real numbers, usually called their X-, Y- and Z-coordinates. To define these coordinates, three perpendicular directed lines (the X-, Y- and Z-axes) joining at a particular point called the *origin of coordinates*, are specified, as well as the unit length, which is marked off on the three axes (Fig. 1.1).

The 3D Euclidean space can be represented by such a Cartesian coordinate frame: every point $p \in \mathbb{E}^3$ can be identified with a point in $\mathbb{R}^3$ with three coordinates

$$\mathbf{p} \triangleq \begin{bmatrix} x & y & z \end{bmatrix}^\top = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \in \mathbb{R}^3 .$$

The definition of a metric is very closely related to the concept of vector. A *vector* can be defined as a directed arrow connecting p to a second point q, and denoted $v \triangleq \overrightarrow{pq}$. In $\mathbb{E}^3$ its

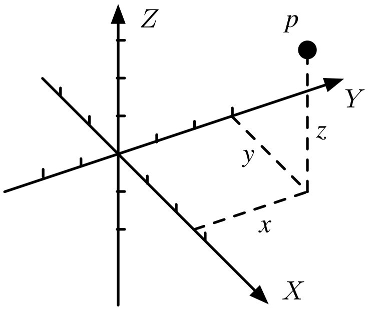

Figure 1.1: A three-dimensional Cartesian coordinate frame.

coordinates $\mathbf{v} = [v_x, v_y, v_z]^\top$ are defined as

$$\mathbf{v} \triangleq \mathbf{q} - \mathbf{p} \quad \in \mathbb{R}^3,$$

where $\mathbf{p}$ and $\mathbf{q}$ are the coordinates of p and q respectively. As this vector is glued to both extreme points, we can refer to it with the term *bounded vector*. Of greater convenience is the notion of *free vector*: we say that two pairs of points (p, q) and (p', q') define the same free vector if their coordinates satisfy $\mathbf{q} - \mathbf{p} = \mathbf{q}' - \mathbf{p}'$. Normally and as we will do in this work, free vectors are simply called *vectors*. The set of all vectors forms a *linear vector space*, that is, any linear combination of vectors results in a new vector:

$$\alpha\mathbf{v} + \beta\mathbf{u} = [\alpha v_x + \beta u_x, \alpha v_y + \beta u_y, \alpha v_z + \beta u_z]^\top \in \mathbb{R}^3, \quad \forall \alpha, \beta \in \mathbb{R}.$$

Finally, the *Euclidean metric* for $\mathbb{E}^3$ is defined by the *inner product* on the vector space $\mathbb{R}^3$:

$$\langle \mathbf{u}, \mathbf{v} \rangle \triangleq \mathbf{u}^\top \mathbf{v} = u_x v_x + u_y v_y + u_z v_z \in \mathbb{R}, \quad \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^3.$$

Now see how this metric works. The *norm* or length of a vector $\mathbf{v}$ is $\|\mathbf{v}\| \triangleq \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{v_x^2 + v_y^2 + v_z^2}$. The angle α between two vectors is obtained from $\langle \mathbf{u}, \mathbf{v} \rangle = \|\mathbf{u}\| \|\mathbf{v}\| \cos(\alpha)$. When two vectors $\mathbf{u}$ and $\mathbf{v}$ satisfy $\langle \mathbf{u}, \mathbf{v} \rangle = 0$ they are said to be *orthogonal*. From these distances and directions to the more elaborated concepts of sizes and shapes there is just a sequence of trivial steps.

Summarizing, the Euclidean space $\mathbb{E}^3$ can be defined as a space that, with respect to a certain Cartesian frame, can be identified with $\mathbb{R}^3$ and has a metric on its vector space defined by the above inner product. Now we can relax and continue to think about $\mathbb{R}^3$ as an identification of the 3D world: the Cartesian frame is already natural for us, as it is the notion of vector, and the inner product provides all we need to calculate distances and directions.

1.2 Rigid body motions

1.2.1 Frame conventions and notation

The definition of the Euclidean space above contains two important ambiguities that relate to the position and orientation of the three coordinate axes of the Cartesian frame. These ambiguities are not fundamental and should be resolved by an arbitrary choice of the user.

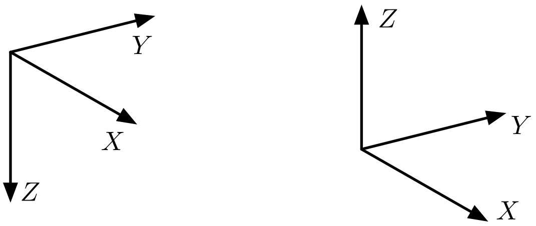

Figure 1.2: Left- and right-handed Cartesian coordinate frames.

First, one can distinguish between two types of 3D coordinate frames: the *left-handed* and the *right-handed* (Fig. 1.2). If you align your thumb, fore-finger and middle-finger respectively with the X-, Y- and Z-axes, you will obtain one of these two possibilities depending on whether you are using your left or your right hand. The idea is that it is not possible to rigidly move one frame to obtain the other one, and thus a choice must be done from the start if we don't want to put into compromise the whole mathematical material that will be introduced soon.

Second, the frame's position and orientation must be clearly specified so that every point in the space can be uniquely identified with respect to it. Different frames will be defined and coexist within complex systems, and tools will be given in the next section to specify the position of points in a particular frame when they are originally expressed in another one.

Additionally, a convenient notation must be defined to clearly refer to frames within the text, or to indicate in which frame the coordinates of a particular point or vector are expressed. We will use calligraphic symbols such as $\mathcal{C}$, $\mathcal{F}$, $\mathcal{R}$ or $\mathcal{W}$, that indicate the frame's nature. Consider two generic frames $\mathcal{F}$ and $\mathcal{G}$: the same point p will have coordinates $\mathbf{p}^{\mathcal{F}}$ when expressed in frame $\mathcal{F}$, and $\mathbf{p}^{\mathcal{G}}$ in frame $\mathcal{G}$. Mobile frames will be denoted $\mathcal{F}(t)$. The frame orientation will also be included in the notation with three additional letters enclosed in brackets and denoting, by order, the directions of the three coordinate axes. For example, $\mathcal{W}\{NWU\}$ would be a world frame with the X-axis pointing 'N'orthwards, the Y-axis pointing 'W'estwards and the Z-axis pointing 'U'pwards.

The conventions adopted in this work are as follows (Fig. 1.3 and Table 1.1):

- **All frames.** All frames are defined right handed.

- **Global or world frame.** Denoted $\mathcal{W}$, this is by definition a unique, static and absolute frame. All other frames are directly or indirectly specified with respect to this one. For convenience, we will try to align the Z-axis with the vertical direction, the other two being left to define the horizontal plane. If one particular direction on this horizontal plane needs to be used as a reference (for example the North direction), it will be aligned with the X-axis. This case would lead to $\mathcal{W}\{NWU\}$. Alternatively, if no information is available about the external, real world, we will simply identify the global frame with the local one (see below) of an object of our interest at the beginning of each experiment. This will be denoted by $\mathcal{W}\{\mathcal{O}(0)\}$, where $\mathcal{O}(0)$ is the frame of object $\mathcal{O}$ at $t = 0$.

- **Local or object frames.** Local frames will be rigidly attached to objects. When these objects are recognizable, we will use this knowledge to align the frame in a suitable way. We have chosen the widely adopted $\{FLU\}$ convention of aligning the frame with the

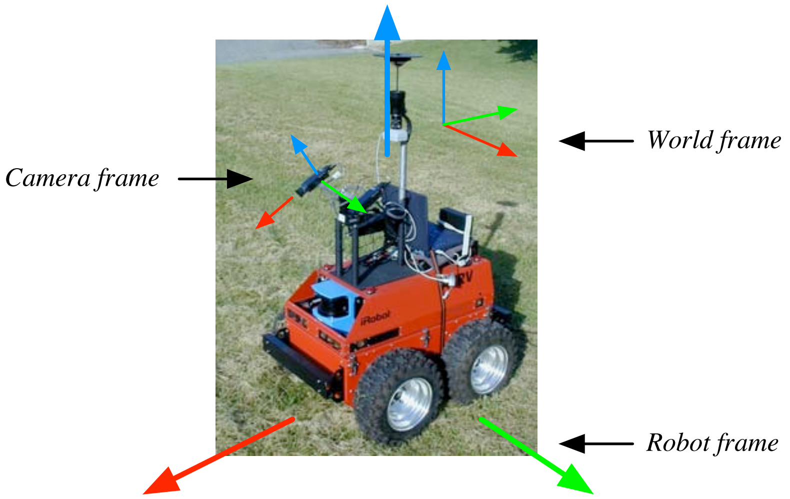

Figure 1.3: Coexisting coordinate frames within a single project. For each frame, X-, Y- and Z-axes are colored red, green and blue respectively. They all correspond to the FLU convention.

X-axis pointing Forward of the object, the Y-axis Leftwards and the Z-axis Upwards. This way, a generic object frame is $\mathcal{O}\{FLU\}$.

- **Sensor frames.** A special case of local frame is the sensor frame. It is specifically thought for cameras, a device that maps information of the 3D world into 2D images. Therefore, such a sensor must specify both the 3D and the 2D axes. We use the notation $\mathcal{S}\{RDF, RD\}$ to indicate that the 3D frame of the sensor is aligned as $\{RDF\}$ (X-Right, Y-Down, Z-Front), while the 2D image is $\{RD\}$. When referring to these frames separately, we will feel free to denote them as sensor $\mathcal{S}\{RDF\}$ and image $\mathcal{I}\{RD\}$. This will be better explained and justified in Chapter 2.

Table 1.1: Summary of frame names and alignments adopted for this work.

Name	Notation	Alignment	Comments
World	$\mathcal{W}$	$\{NWU\}$	
World	$\mathcal{W}$	$\{\mathcal{R}(0)\}$	Same as Robot at $t = 0$
Frame	$\mathcal{F}, \mathcal{G}, \mathcal{H}$	$\{FLU\}$	Generic frames
Object	$\mathcal{O}$	$\{FLU\}$	
Mobile object	$\mathcal{O}(t)$	$\{FLU\}$	
Robot	$\mathcal{R}(t)$	$\{FLU\}$	
Camera	$\mathcal{C}$	$\{FLU\}$	
Sensor	$\mathcal{S}$	$\{RDF; RD\}$	Image frame: u-Right v-Down

1.2.2 Frame transformations

The position and orientation of a frame $\mathcal{F}$ (for *Frame*) relative to another one $\mathcal{W}$ (for *World*) can be precisely specified by the couple $\{\mathbf{t}, \mathbf{R}\}$. In this couple, the vector $\mathbf{t}$ is the position in frame $\mathcal{W}$ of the origin of coordinates of frame $\mathcal{F}$, and is named the *translation vector*. The matrix $\mathbf{R}$ is the *rotation matrix* describing the orientation of frame $\mathcal{F}$ with respect to frame $\mathcal{W}$.

The rotation matrix and translation vector

The interpretation of the translation vector is quite obvious. On the contrary, that of the rotation matrix has received much attention, in my opinion without the necessary success: myself and most of my colleagues are still insecure on how to rapidly construct, interpret or manipulate rotation matrices without having to numerically verify if what we did is correct. Very often we try and, on failure, we transpose the matrix hoping that this will solve the problem. I find this unacceptable and sad, and thus I'm obliged to give an extra word on this —even at the risk of failing to clarify things up.

The purpose of the rotation matrix is to rotate a vector $\mathbf{v}$ by means of a linear operation $\mathbf{w} = \mathbf{R}\,\mathbf{v}$. This is equivalent to expressing the same vector v in two different reference frames $\mathcal{W}$ and $\mathcal{F}$ so that $\mathbf{v}^{\mathcal{W}} = \mathbf{R}\,\mathbf{v}^{\mathcal{F}}$, where the frame $\mathcal{F}$ is a rotated version of $\mathcal{W}$ by an amount specified by $\mathbf{R}$. With this in mind, the intuitive form of defining the rotation matrix that I like the most is given in the next proposition, is demonstrated afterwards, and is illustrated in Fig. 1.4:

Proposition 1.1 (Rotation matrix). *The rotation matrix that expresses the orientation of a frame $\mathcal{F}$ with respect to a frame $\mathcal{W}$ is a square, orthonormal matrix. Its columns are the vectors, expressed in $\mathcal{W}$ coordinates, of the canonical orthonormal basis of $\mathcal{F}$.* □

Proof Consider the canonical orthonormal basis of $\mathcal{F}$:

$$\{\mathbf{e}_1, \cdots, \mathbf{e}_n\} = \{[1, 0, \cdots, 0]^\top, [0, 1, \cdots, 0]^\top, \cdots, [0, \cdots, 0, 1]^\top\}.$$

The same vectors written in $\mathcal{W}$ coordinates form the non-canonical orthonormal basis of $\mathcal{W}$:

$$\{\mathbf{r}_1, \cdots, \mathbf{r}_n\}.$$

Any vector v in $\mathcal{F}$ is a linear combination $\mathbf{v}^{\mathcal{F}} = v_1\mathbf{e}_1 + \cdots + v_n\mathbf{e}_n = [v_1, \cdots, v_n]^\top$ of the vectors of the canonical basis. In $\mathcal{W}$ the vector is the same linear combination of the vectors of the basis of $\mathcal{W}$, *i.e.* $\mathbf{v}^{\mathcal{W}} = v_1\mathbf{r}_1 + \cdots + v_n\mathbf{r}_n$. This is written in matrix form as

$$\mathbf{v}^{\mathcal{W}} = \begin{bmatrix}\mathbf{r}_1 & \cdots & \mathbf{r}_n\end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}.$$

Defining $\mathbf{R} = [\mathbf{r}_1 \cdots \mathbf{r}_n]$ the above expression becomes

$$\mathbf{v}^{\mathcal{W}} = \mathbf{R}\,\mathbf{v}^{\mathcal{F}},$$

where the matrix $\mathbf{R}$ is the rotation matrix. ■

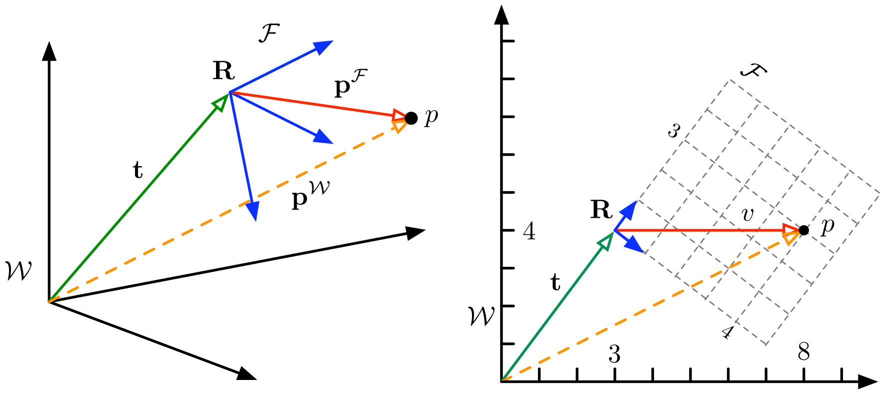

Figure 1.4: Frame transformation in 3D and 2D. In 2D, the known fact $3^2 + 4^2 = 5^2$ is used to visually demonstrate the rotation matrix construction and operation: vector $\mathbf{t} = [3, 4]^\top$ is length 5, as well as vector v that has coordinates $\mathbf{v}^\mathcal{F} = [4, 3]^\top$ and $\mathbf{v}^\mathcal{W} = [5, 0]^\top$; the canonical orthonormal basis of $\mathcal{F}$ is formed by the two blue vectors, that become $\mathbf{r}_1 = [4/5, -3/5]^\top$ and $\mathbf{r}_2 = [3/5, 4/5]^\top$ in $\mathcal{W}$ frame. The rotation matrix is formed from these two unit vectors, that is $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2]$. You can easily verify that equations 1.5 and 1.6 hold for point p.

By construction, the rotation matrix is an orthonormal matrix with the following properties:

$$
\begin{aligned}
\mathbf{R}^{-1} &= \mathbf{R}^\top && (1.1)\\
\det(\mathbf{R}) &= 1 && (1.2)\\
\|\mathbf{R}\,\mathbf{v}\| &= \|\mathbf{v}\| && (1.3)\\
\langle \mathbf{R}\,\mathbf{u}, \mathbf{R}\,\mathbf{v}\rangle &= \langle \mathbf{u}, \mathbf{v}\rangle . && (1.4)
\end{aligned}
$$

that have the following geometrical interpretation: the inverse rotation is represented by the transposed matrix (1.1); rotations conserve the hand frames are ruled with (1.2); vector lengths are conserved by rotation (1.3); and angles between equally rotated vectors are also conserved (1.4). They as a whole guarantee true rigid body rotations.

The whole 3D frame transformation (Fig. 1.4) includes a translation encoded by vector $\mathbf{t}$ and the rotation $\mathbf{R}$. Identifying the vector $\mathbf{v}^\mathcal{F} \equiv \mathbf{p}^\mathcal{F}$ and remarking that $\mathbf{p}^\mathcal{W} = \mathbf{t} + \mathbf{v}^\mathcal{W}$ we easily get the transformation '*from frame* $\mathcal{F}$ to frame $\mathcal{W}$', that we name $fromFrame(\cdot)$:

$$
\mathbf{p}^\mathcal{W} = fromFrame(\mathcal{F}, \mathbf{p}^\mathcal{F}) \triangleq \mathbf{R}\mathbf{p}^\mathcal{F} + \mathbf{t}. \tag{1.5}
$$

The opposite transformation 'from frame $\mathcal{W}$ *to frame* $\mathcal{F}$' results from inverting the previous expression and is denoted $toFrame(\cdot)$:

$$
\mathbf{p}^\mathcal{F} = toFrame(\mathcal{F}, \mathbf{p}^\mathcal{W}) \triangleq \mathbf{R}^\top\mathbf{p}^\mathcal{W} - \mathbf{R}^\top\mathbf{t}. \tag{1.6}
$$

Remark 1.1 (Transforming vectors). By defining a vector $\mathbf{v}$ as a difference of two points $\mathbf{v} \triangleq \mathbf{q} - \mathbf{p}$ we can write the last transformations applied to vectors simply as

$$
\begin{aligned}
\mathbf{v}^\mathcal{W} &= fromFrame(\mathcal{F}, \mathbf{q}^\mathcal{F}) - fromFrame(\mathcal{F}, \mathbf{p}^\mathcal{F}) &&= \mathbf{R}\,\mathbf{v}^\mathcal{F}\\
\mathbf{v}^\mathcal{F} &= \quad toFrame(\mathcal{F}, \mathbf{q}^\mathcal{W}) - toFrame(\mathcal{F}, \mathbf{p}^\mathcal{W}) &&= \mathbf{R}^\top\,\mathbf{v}^\mathcal{W},
\end{aligned}
$$

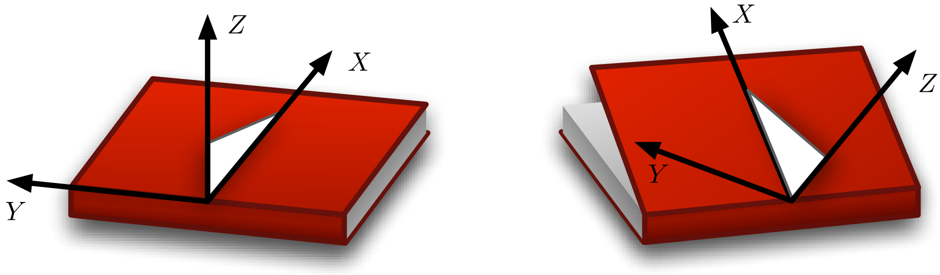

Figure 1.5: The Euler angles defined in the ZYX convention. Three consecutive rotations transform the initial frame (*left*) to the rotated frame (*right*). Put a book on the table and set the $\{FLU\}$ axes as drawn, with a vertical triangle at the XZ plane. Now *1)* turn the book counterclockwise an angle ψ (yaw: around the Z-axis). Then *2)* open the cover an angle $-\theta$ (negative pitch, around the new Y-axis). Finally *3)* incline the triangle to the right an angle ϕ (roll, around the new X-axis). You get the fully 3D rotated frame of the right-hand sketch.

i.e. vectors are invariant upon translation, as they must be. To remark vector operation we use a bar over the function names as follows:

$$\mathbf{v}^{\mathcal{W}} = \overline{fromFrame}(\mathcal{F}, \mathbf{v}^{\mathcal{F}}) \triangleq \mathbf{R}\,\mathbf{v}^{\mathcal{F}} \tag{1.7}$$

$$\mathbf{v}^{\mathcal{F}} = \overline{toFrame}(\mathcal{F}, \mathbf{v}^{\mathcal{W}}) \triangleq \mathbf{R}^{\top}\,\mathbf{v}^{\mathcal{W}}. \tag{1.8}$$

□

Remark 1.2 (Minimally encoded rotations). The rotation matrix is very convenient to perform rotations in 3D, but it is not so indicated for actually encoding them because of its enormous redundancy: while the rotation matrix contains nine entries, any rotation in 3D can be specified with just three parameters. □

In this work, two other representations are used together with the rotation matrix to encode orientations: the *Euler angles* and the *quaternion*. We give now a brief description of what is relevant to us about them and how they relate to the rotation matrix.

The Euler angles

Generally speaking, any orientation in 3D can be obtained with three consecutive rotations around three different axes. The three rotated angles are commonly referred to as the *Euler angles*. Several conventions exist (with no real consensus[1]) on the specification of these axes and on the order in which the rotations are performed. In this work we take the ZYX convention (Fig. 1.5) that corresponds to a first rotation around the Z axis (*yaw* angle), a second rotation around the rotated Y axis (*pitch* angle) and a third one around the doubly rotated X axis (*roll* angle).

Euler angles are normally defined for aircraft, where *pitch* is defined as positive when the vehicle's nose raises. This is therefore a definition related to a reference frame of the $\{FRD\}$ type[2]. However, it can be extended to any other frame as long as we maintain the three

[1] It is interesting to read the Euler angles entry in `mathworld.wolfram.com`.

[2] In $\{FRD\}$, a positive rotation around the Y-axis, which is at the right wing, raises the plane's tip.

rotations in the order ZYX. For example, in an $\{FLU\}$ frame, which is more common for terrestrial vehicles, positive *pitch* means that the vehicle's nose is sinking because of a positive rotation around the (left sided!) Y-axis.

As $\{FLU\}$ is the convention adopted for all mobile objects in this work, let us put things clear once and for all. See table 1.2.

Table 1.2: The Euler angles in $\{FLU\}$ frames.

Euler angle	axis	positive means...
Yaw (ψ)	vertical (Z)	turn left
Pitch (θ)	transversal (Y)	nose sinks
Roll (ϕ)	longitudinal (X)	sway, right side sinks

Given the Euler angles $\mathbf{e} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\top$ corresponding to the *roll*, *pitch* and *yaw* angles of the attitude of a reference frame, the rotation matrix is obtained by performing the product of the three rotation matrices corresponding to the three elementary rotations. We get:

$$\mathbf{R}(\mathbf{e}) = \mathrm{e2R}(\mathbf{e}) \triangleq \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{1.9}$$

The Euler angles constitute a minimal representation for rotations. They have the disadvantage of presenting discontinuities, so when angles evolve one has to take care to bring them to their convenient ranges:

$$\phi \in \begin{bmatrix}-\pi & \pi\end{bmatrix}, \qquad \theta \in \begin{bmatrix}-\pi/2 & \pi/2\end{bmatrix}, \qquad \psi \in \begin{bmatrix}0 & 2\pi\end{bmatrix}.$$

The quaternion

A quaternion is a generalization of the notion of complex number: if C and D are two complex numbers of the form $C = a + bi \in \mathbb{C}$, $D = c + di \in \mathbb{C}$; $\forall a, b, c, d \in \mathbb{R}$, where i is defined so that $i^2 \triangleq -1$, then we define the quaternion $Q \in \mathbb{H}$ by

$$Q \triangleq C + Dj \in \mathbb{H}; \qquad \forall C, D \in \mathbb{C} \tag{1.10}$$

where j is defined so that $j^2 \triangleq -1$ and $ij \triangleq -ji \triangleq k$. These definitions fix the necessary algebra in the space of quaternions $\mathbb{H}$. We can trivially develop (1.10) to obtain $Q = a + bi + cj + dk$. A quaternion Q will be often written in the form of a vector $\mathbf{q}$:

$$\mathbf{q} = [a, b, c, d]^\top \in \mathbb{R}^4.$$

The conjugated of the quaternion is defined by $\mathbf{q}^* = [a, -b, -c, -d]^\top$ and its norm $\|\mathbf{q}\| \triangleq \sqrt{\mathbf{q} \cdot \mathbf{q}^*}$, where the explicit writing of the multiplication dot $\cdot$ indicates the use of quaternion algebra. If $\|\mathbf{q}\| = 1$ the quaternion is said to be *normalized*.

Interesting enough is the fact that, while normalized complex numbers can be used to encode any rotation in the Euclidean plane, normalized quaternions can play the same role in the Euclidean 3D space. Indeed, consider a normalized quaternion, which can always be written as

$$\mathbf{q} = \begin{bmatrix} \cos\theta \\ u_x \sin\theta \\ u_y \sin\theta \\ u_z \sin\theta \end{bmatrix}. \tag{1.11}$$

where $[u_x, u_y, u_z]^\top = \mathbf{u}^\top$ is a unit vector $\|\mathbf{u}\| = 1$ and θ is a real scalar. Expressing a generic vector $\mathbf{v} \in \mathbb{E}^3$ in the quaternion space $\mathbb{H}$ as $\widetilde{\mathbf{v}} = 0 + i\,v_x + j\,v_y + k\,v_z \in \mathbb{H}$, the operation

$$\widetilde{\mathbf{w}} = \mathbf{q} \cdot \widetilde{\mathbf{v}} \cdot \mathbf{q}^*$$

performs to vector $\mathbf{v}$ a rotation of θ radians around the axis defined by $\mathbf{u}$. Writing this expression linearly in $\mathbf{v}$ leads to its equivalent matrix form $\mathbf{w} = \mathbf{R}\,\mathbf{v}$ which defines the rotation matrix $\mathbf{R}$ as a function of the quaternion $\mathbf{q} = [a, b, c, d]^\top$:

$$\mathbf{R}(\mathbf{q}) = \mathrm{q2R}(\mathbf{q}) \triangleq \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2 \cdot (bc - ad) & 2 \cdot (bd + ac) \\ 2 \cdot (bc + ad) & a^2 - b^2 + c^2 - d^2 & 2 \cdot (cd - ab) \\ 2 \cdot (bd - ac) & 2 \cdot (cd + ab) & a^2 - b^2 - c^2 + d^2 \end{bmatrix}. \tag{1.12}$$

The opposite rotation is obtained with the conjugate quaternion $\mathbf{q}^*$ so that $\mathbf{R}(\mathbf{q}^*) = \mathbf{R}^\top(\mathbf{q})$. The negated quaternion represents exactly the same rotation as the original one, that is $\mathbf{R}(-\mathbf{q}) = \mathbf{R}(\mathbf{q})$. This shows that the quaternion is a non minimal representation for rotations, and therefore the normalization constraint must be imperatively insured after any modification. Rotation compositions satisfy $\mathbf{R}(\mathbf{q}_1 \cdot \mathbf{q}_2) = \mathbf{R}(\mathbf{q}_1)\mathbf{R}(\mathbf{q}_2)$.

Another useful identity that we will use when introducing dynamic models in Section 1.3 is the derivative of the quaternion. This will be related to the *angular velocity* $\boldsymbol{\omega}$ defined in the rotating frame as

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

Performing temporal differentiation of (1.11) we can show after tedious calculations that the following identity holds

$$2\mathbf{q}^* \cdot \dot{\mathbf{q}} = \widetilde{\boldsymbol{\omega}}$$

where $\widetilde{\boldsymbol{\omega}} = [0, \omega_x, \omega_y, \omega_z]^\top$. This and the fact $\mathbf{q} \cdot \mathbf{q}^* = 1$ lead to

$$\dot{\mathbf{q}} = \frac{1}{2}\,\mathbf{q} \cdot \widetilde{\boldsymbol{\omega}}$$

which, being linear in $\mathbf{q}$, permits us to write the time-derivative of a quaternion in linear (matrix) form:

$$\dot{\mathbf{q}} = \frac{1}{2}\,\boldsymbol{\Omega}\,\mathbf{q}, \tag{1.13}$$

where $\boldsymbol{\Omega}$ is a skew symmetric matrix built from the components of $\boldsymbol{\omega}$:

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \tag{1.14}$$

1.2.3 Manipulating different rotation representations in a unique project

We have seen three different representations for rotations so far: the Euler angles, the quaternion and the rotation matrix. All forms have their pros and cons, and a brief discussion follows to justify their usage (Fig. 1.6).

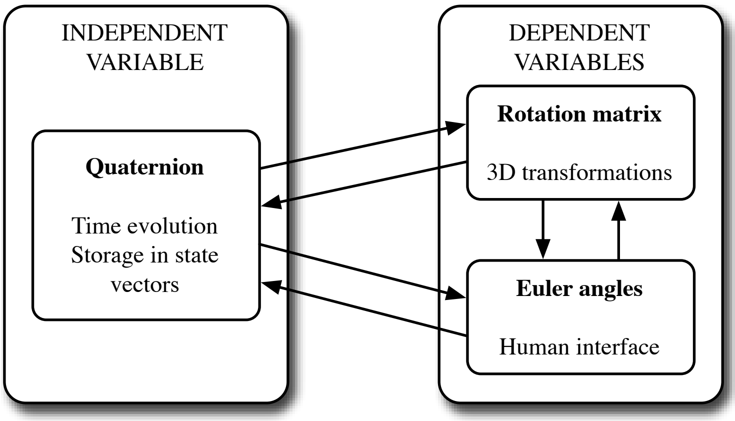

Figure 1.6: Main and derived rotation representations.

The **rotation matrix** is used to perform the rotations to the points and vectors in 3D space. The reason is that the algebra is straightforward and linear, and that the number of computing operations is minimized.

The **quaternion** is used to store orientation information in the state vectors. The time evolution equations based on quaternions are continuous and continuously derivable, something that will be very important for later automatic manipulations such as filtering (which is all I do actually). Because the quaternion $\mathbf{q}$ plays such a central role, it will be convenient to use it by default in all frame specifications. This way, a generic frame $\mathcal{F}$ will be numerically specified by a 7-vector as

$$\mathcal{F} = \begin{bmatrix} \mathbf{t} \\ \mathbf{q} \end{bmatrix}. \tag{1.15}$$

The **Euler angles** are easy to visualize and to be understood by a human user[3], and are often used to provide input and output human interface. The pose of a camera in a robot is easily specified this way. They are also useful in odometry as the relatively small changes in vehicle orientation are easily expressed in Euler angles. And in inertial measurement units (IMU), the set of three orthogonal gyrometers will directly provide the three Euler angular rates $\boldsymbol{\omega}$ in robot frame —precisely those appearing in the derivative of the quaternion.

In order to facilitate all possible passages between these three representations, six conversion functions must be defined. They can be found in Appendix C.

1.2.4 The homogeneous matrix

Frame transformations (1.5,1.6) belong to the affine transformations group. They can be embedded into the linear transformations group by using *homogeneous coordinates*. We give

[3] We could not do any such things as Table 1.2 and Fig. 1.5 for the quaternion or the rotation matrix.

the following definition:

Definition 1.1 (Homogeneous point). Given a point $p \in \mathbb{E}^n$ with coordinates $\mathbf{p} \in \mathbb{R}^n$, its homogeneous representation is denoted $\underline{\mathbf{p}}$ and defined by

$$\underline{\mathbf{p}} \triangleq \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1}. \qquad \square$$

Now we can write the frame transformation $fromFrame(\cdot)$ linearly with

$$\begin{bmatrix} \mathbf{p}^{\mathcal{W}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}^{\mathcal{F}} \\ 1 \end{bmatrix}$$

and $toFrame(\cdot)$ with

$$\begin{bmatrix} \mathbf{p}^{\mathcal{F}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}^{\mathcal{W}} \\ 1 \end{bmatrix}.$$

The intervening matrices are called the *homogeneous matrices* and are defined as follows:

$$\mathbf{H}^{\mathcal{W}\mathcal{F}} \triangleq \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \qquad \mathbf{H}^{\mathcal{F}\mathcal{W}} \triangleq \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}$$

that obviously satisfy $(\mathbf{H}^{\mathcal{W}\mathcal{F}})^{-1} \equiv \mathbf{H}^{\mathcal{F}\mathcal{W}}$. To illustrate the adopted notation let us put the homogeneous transformations in compact form

$$\underline{\mathbf{p}}^{\mathcal{W}} = \mathbf{H}^{\mathcal{W}\mathcal{F}}\,\underline{\mathbf{p}}^{\mathcal{F}}; \qquad \underline{\mathbf{p}}^{\mathcal{F}} = \mathbf{H}^{\mathcal{F}\mathcal{W}}\,\underline{\mathbf{p}}^{\mathcal{W}}.$$

See how the frame notation superscripts on the homogeneous matrices follow the *right to left* logic to indicate the direction of the transformation.

1.2.5 Composition of transformations

Consider a scenario consisting of a robot, a static object and a moving one. The robot has two fixed cameras mounted somewhere on its chassis, each one with its own sensor.[4] This scenario is depicted in Fig. 1.7.

Consider the case where a point $\mathbf{p}^{\mathcal{O}_1}$ belonging to object $\mathcal{O}_1$ is observed from camera $\mathcal{C}_1$. The coordinates of this point with respect to the sensor frame $\mathcal{S}_1$ (dashed arrow in the figure) are easily obtained by using the homogeneous formulation:

$$\underline{\mathbf{p}}^{\mathcal{S}_1} = \mathbf{H}^{\mathcal{S}_1\mathcal{C}_1}\,\mathbf{H}^{\mathcal{C}_1\mathcal{R}}\,\mathbf{H}^{\mathcal{R}\mathcal{W}}\,\mathbf{H}^{\mathcal{W}\mathcal{O}_1}\,\underline{\mathbf{p}}^{\mathcal{O}_1} \tag{1.16}$$

This same transformation can be written in function form as

$$\mathbf{p}^{\mathcal{S}_1} = toFrame\Big[\mathcal{S}_1, toFrame\big[\mathcal{C}_1, toFrame\big(\mathcal{R}, fromFrame(\mathcal{O}_1, \mathbf{p}^{\mathcal{O}_1})\big)\big]\Big]. \tag{1.17}$$

[4] In cameras, we distinguish between body frame and sensor frame. This will be explained in Chapter 2.

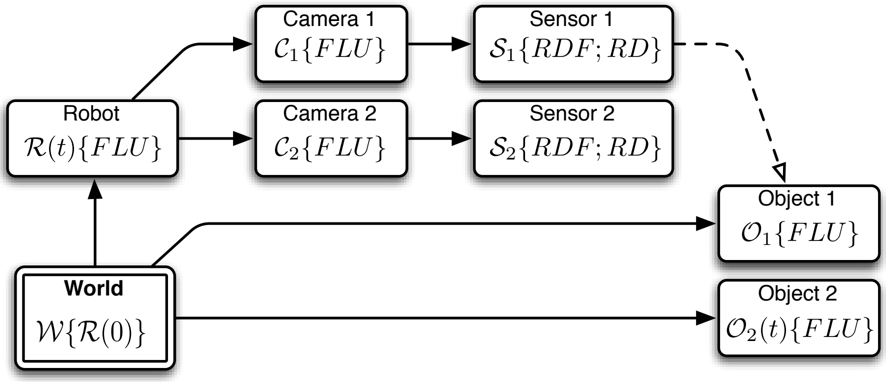

Figure 1.7: Multiple Coordinate frames in a single project. This kind of diagram will be used to illustrate the different kinematic chains used throughout this work.

1.3 Dynamic scenarios

1.3.1 From continuous-time to discrete-time

We start at the state space formulation to represent continuous-time systems such as a moving object or a robot:

$$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = \phi(\mathbf{x}(t), \mathbf{u}(t)) \tag{1.18}$$

where $\mathbf{x}(t)$ is the state vector describing the current state of the system and $\mathbf{u}(t)$ is a vector of controls that permit to modify its trajectory in the state space. In computer-based systems where calculations are performed at discrete time instants $t = k\,\Delta t$, it is necessary to translate this differential equation into a differences one. The approximation used corresponds to

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} \approx \frac{\mathbf{x}(t) - \mathbf{x}(t - \Delta t)}{\Delta t} = \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta t} \tag{1.19}$$

that requires a sampling time Δt small enough compared to the system dynamics: when this time tends to zero the above approximation tends to the definition of the derivative. Using 1.19 the differential equation 1.18 is approximated with

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \Delta t\, \phi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}).$$

Defining $\mathbf{f}(\mathbf{x}, \mathbf{u}) \triangleq \mathbf{x} + \Delta t\, \phi(\mathbf{x}, \mathbf{u})$ we get

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}). \tag{1.20}$$

In this work we have tried to avoid unnecessary notations when the expressions contain no ambiguity. We introduce this lighter notation (the $(\cdot)^+$ notation):

$$\mathbf{x}^+ = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{1.21}$$

where $\mathbf{x}^+$ means *the updated value of* $\mathbf{x}$, valid for any $\mathbf{x}$ and for any kind of update —in this case a time update.

1.3.2 Odometry models

As the robot moves, its pose $\mathcal{R}^\top = [\mathbf{x}^\top \; \mathbf{q}^\top]$ evolves following a particular model

$$\mathcal{R}^+ = \mathbf{f}(\mathcal{R}, \mathbf{u}) \tag{1.22}$$

where $\mathbf{u}$ is a vector of controls or odometry measurements.

The form of the evolution function $\mathbf{f}$ depends on the format of the control or odometry data $\mathbf{u}$. The algebra for two different odometry formats is presented below.

Position and Euler increments in robot frame

The odometry vector is in the form

$$\mathbf{u} = \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{e} \end{bmatrix} = \begin{bmatrix} \delta x \\ \delta y \\ \delta z \\ \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix} \tag{1.23}$$

where $\delta\mathbf{x}$ is the increment in the robot position and $\delta\mathbf{e}$ is the increment in the robot orientation expressed in Euler angles, all in $\mathcal{R}$ frame. The evolution functions are

$$\mathbf{x}^+ = \mathbf{x} + \mathbf{R}(\mathbf{q}) \cdot \delta\mathbf{x} = \mathit{fromFrame}(\mathcal{R}, \delta\mathbf{x}) \tag{1.24}$$

$$\mathbf{q}^+ = \mathbf{q} + \frac{1}{2}\mathbf{\Omega}(\delta\mathbf{e}) \cdot \mathbf{q} \tag{1.25}$$

where $\mathbf{R}(\mathbf{q})$ is the rotation matrix corresponding to the orientation $\mathbf{q}$ and $\mathbf{\Omega}(\delta\mathbf{e})$ is the skew symmetric matrix (already defined in (1.14))

$$\mathbf{\Omega}(\delta\mathbf{e}) = \begin{bmatrix} 0 & -\delta\phi & -\delta\theta & -\delta\psi \\ \delta\phi & 0 & \delta\psi & -\delta\theta \\ \delta\theta & -\delta\psi & 0 & \delta\phi \\ \delta\psi & \delta\theta & -\delta\phi & 0 \end{bmatrix}$$

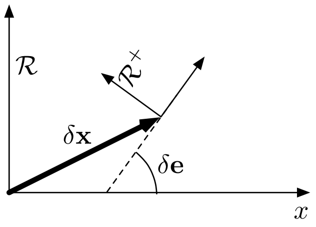

Figure 1.8: Odometry model with position and Euler increments.

Forward motion and Euler increments in robot frame

For non-holonomic robots this is an interesting representation because it contains only a forward motion and the orientation increments, leading to an odometry vector of only four components. For a reference frame with the x axis looking forward like $\{FLU\}$ we have

$$\mathbf{u} = \begin{bmatrix} \delta x \\ \delta \mathbf{e} \end{bmatrix} = \begin{bmatrix} \delta x \\ \delta \phi \\ \delta \theta \\ \delta \psi \end{bmatrix} \tag{1.26}$$

where δx is the forward motion and $\delta \mathbf{e}$ is the increment in the robot orientation expressed in Euler angles, all in $\mathcal{R}$ frame. The evolution functions are

$$\mathbf{x}^+ = \mathbf{x} + \mathbf{R}(\mathbf{q} + \frac{1}{4}\boldsymbol{\Omega}(\delta \mathbf{e}) \cdot \mathbf{q}) \cdot \begin{bmatrix} \delta x \\ 0 \\ 0 \end{bmatrix} \tag{1.27}$$

$$\mathbf{q}^+ = \mathbf{q} + \frac{1}{2}\boldsymbol{\Omega}(\delta \mathbf{e}) \cdot \mathbf{q} \tag{1.28}$$

where $\mathbf{R}(\mathbf{q} + \frac{1}{4}\boldsymbol{\Omega}(\delta \mathbf{e}) \cdot \mathbf{q})$ is the rotation matrix corresponding to the robot orientation $\mathbf{q}$ plus one half of the orientation increment $\delta \mathbf{e}$. This corresponds to uniformly distributing the full rotation $\delta \mathbf{e}$ over the full displacement δx as illustrated in Fig. 1.9.

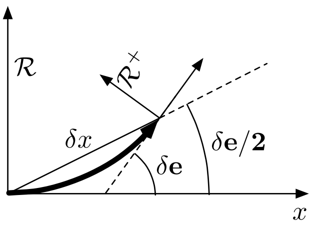

Figure 1.9: Odometry model with forward motion and Euler increments.

1.3.3 Dynamic models

We propose the kinematics functions of more complete dynamic models, specially for constant-velocity models or constant-acceleration models. They are better suited for objects over which we do not have any control. They simply try to reproduce the physics laws of inertia (Newton's first and second laws).

Constant velocity dynamic model in world frame

The evolution model for the robot state vector $\mathcal{R}^\top = [\mathbf{x}^\top, \mathbf{v}^\top, \mathbf{q}^\top, \omega^\top]$ is written as

$$\begin{aligned} \mathbf{x}^+ &= \mathbf{x} + T_s \cdot \mathbf{v} \\ \mathbf{v}^+ &= \mathbf{v} + \upsilon_\mathbf{v} \\ \mathbf{q}^+ &= \mathbf{q} + \frac{1}{2} T_s \boldsymbol{\Omega}(\omega) \cdot \mathbf{q} \\ \omega^+ &= \omega + \upsilon_\omega \end{aligned} \tag{1.29}$$

where T_s is the sampling time, $\mathbf{x}$ and $\mathbf{q}$ are the position and the orientation quaternion, $\mathbf{v}$ and ω are the linear and angular velocities, and $\upsilon_\mathbf{v}$ and υ_ω are independent velocity perturbations,[5]

[5] We will learn in Chapter 3 how to define them as stochastic processes such as white Gaussian noises.

i.e. the forces in the Newtonian sense. All is expressed in world frame except angular velocity ω which is in robot frame.

Constant acceleration dynamic models in world frame

In Inertial aided systems we dispose of measures of acceleration and angular velocities. It is convenient to define the dynamic model with the linear acceleration state included as follows. Consider the robot's state vector $\mathcal{R}^\top = [\mathbf{x}^\top, \mathbf{v}^\top, \mathbf{a}^\top, \mathbf{q}^\top, \omega^\top]$. Its dynamic model is

$$\begin{aligned}
\mathbf{x}^+ &= \mathbf{x} + T_s \cdot \mathbf{v} + \frac{1}{2} T_s^2 \cdot \mathbf{a} \\
\mathbf{v}^+ &= \mathbf{v} + T_s \cdot \mathbf{a} \\
\mathbf{a}^+ &= \mathbf{a} + \upsilon_\mathbf{a} \\
\mathbf{q}^+ &= \mathbf{q} + \frac{1}{2} T_s \mathbf{\Omega}(\omega) \cdot \mathbf{q} \\
\omega^+ &= \omega + \upsilon_\omega
\end{aligned}$$

where $\upsilon_\mathbf{a}$ is the perturbation affecting the linear acceleration.

In such systems, accelerations and angular rates measurements are provided by sensors which often exhibit important drifts [Grewal and Andrews 1993]. It is common to self-calibrate these drifts by *filtering*[6] the system

$$\begin{aligned}
\mathbf{x}^+ &= \mathbf{x} + T_s \cdot \mathbf{v} + \frac{1}{2} T_s^2 \cdot \mathbf{a} \\
\mathbf{v}^+ &= \mathbf{v} + T_s \cdot \mathbf{a} \\
\mathbf{a}^+ &= \mathbf{a} + \upsilon_\mathbf{a} \\
\mathbf{q}^+ &= \mathbf{q} + \frac{1}{2} T_s \mathbf{\Omega}(\omega) \cdot \mathbf{q} \\
\omega^+ &= \omega + \upsilon_\omega \\
\alpha^+ &= \alpha \\
\gamma^+ &= \gamma
\end{aligned}$$

where α and γ are the drifts of the accelerometer and the gyrometer, initially unknown, which are supposed to be constant and which must explicitly appear in the observation model equations —and they must further be observable, of course! These solutions are beyond the scope of this work.

[6] Filtering, as in Chapter 3.

Chapter 2

Vision

> Either to make tools to enhance our natural abilities (thus becoming, to a certain extent, *superhumans*[1]), or to provide some answer to the question of *who we are* (thus becoming, to a certain extent, *Gods*), men have since long ago tried to reinvent nature.

2.1 Introduction

Computer vision is a new discipline in this vast history of men's wills. Long before computers were imagined, common sense suggested that men should try to build an eye before attempting with a brain. So we did. Later and inevitably, we started thinking about building brains. Today, with modern digital video cameras, we have built (very good) artificial eyes that can be connected to (far less good) artificial brains, the computers. Today, reinventing vision is not a matter of building a device that *captures* the visible world but one that *understands* what is captured.

This chapter revises the image formation process of real perspective cameras, with its direct and inverse mathematical models. The material included here is well known and can also be found in excellent works like [Hartley and Zisserman 2000; Horaud and Monga 1995; Ma et al. 2004; Faugeras 1993]. However, during my progressive insight into the subject I found that some approaches were not properly justified from the physical point of view. Ma et al. [2004], for instance, introduce the thin lens camera and then they simplify it to present, afterwards, the pin-hole camera. The pin-hole equations are then introduced and the uncurious engineer can feel satisfied, because one can actually work with these equations. However, a curious mind may notice that the original camera is actually the pin-hole one, but as it can not work in the real world, we are obliged to introduce the lens, which has plenty of undesired side-effects and imperfections that have to be modeled. I tried to take an approach which can be justified in the real world, not only in the engineers world.

I also put special attention to the phenomenon of lens distortion, specially in the way to correct it, and a new method to inverse the distortion function is given. This will be very useful to process only the information of a sparse set of points of the image in order to reconstruct the properties of the external world, as we will do in all the following chapters of this thesis.

The chapter also includes a brief introduction to some well known (automatic) image processing techniques, notably in features detection and matching. It does so up to the extent

[1] Have you read Schrödinger's little book *'Nature and the Greeks'*?

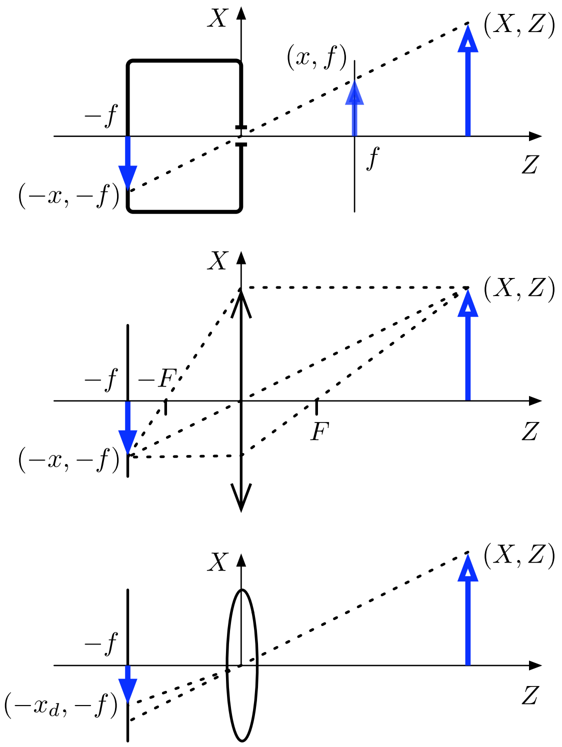

Figure 2.1: Modelling of a perspective camera. Light traveling in straight lines suffices to understand the Pin-Hole camera principle (*top*). An ideal, focused convergent lens accomplishes the same goal, with an enormous gain in luminosity (*center*). Distortion induced by real lenses must be taken into account (*bottom*).

needed for the purposes of this thesis, which wants to contribute a further step towards the ability of men-made machines to visually understand their surroundings.

2.2 Perspective cameras: a geometrical model

Fig. 2.1 depicts in a simplified 2D world the basic steps needed to build a reasonably accurate model of a real perspective camera: *a)* the pin-hole camera, which serves as the base model; *b)* the ideal single-lens camera; and *c)* the real single-lens camera. Real, multi-lens cameras are not studied as they don't contribute to the comprehension of real image formation –they just resolve technological issues of secondary order.

2.2.1 The pin-hole camera model

A pin-hole camera (Fig. 2.1 *top*) consists of an axis, named *optical axis*, a plane perpendicular to the axis, named *focal plane*, with an infinitely small hole situated at the *optical center*, *i.e.* the intersection of the plane with the axis. A second parallel plane, named *image plane*, is situated behind the focal plane at a distance f, called the *focal distance*. The point where the optical axis intersects the image plane is called the *principal point*.

In a pin-hole camera, light rays coming from the external 3D world are projected into the 2D image plane in a straight line through the optical center. This geometrical operation named *projection* may be viewed as an injective application P that maps points in the 3D space into points in the 2D plane:

$$P : \mathbb{R}^3 \rightarrow \mathbb{R}^2; \quad (X, Y, Z) \mapsto (x, y) = P(X, Y, Z). \tag{2.1}$$

In this plane, the image can similarly be viewed as another injective application I that maps every point in the plane into a generic vector defining its photometric properties:

$$I : \mathbb{R}^2 \rightarrow \mathbb{R}^n; \quad (x, y) \mapsto I(x, y). \tag{2.2}$$

Obviously, the photometric properties of each point in the image are strongly related to those existing at the 3D source point, which in turn depend on the scene lightning conditions (spot and diffuse lights, light intensity and temperature, etc.) and the material properties (Lambertian[2], specular, translucent, transparent surfaces). Also, the produced image depends on the properties of the sensing material in the image plane (color, black-and-white, infra-red images). This strong relation between a scene and its image will allow us to detect and track objects by accurately analyzing the images they produce. In this work we will just assume that these images exist and that the information they contain *belong to* or *speak about* the external world through some *geometrical* and *photometrical* transformations, and we will leave the study of the photometrical ones behind.

We are more interested in the projection operation which involves the geometric properties of our system. To see how this works, revisit Fig. 2.1 (*top*) and identify the Z-axis with the optical axis: a light ray coming from a point (X, Z) enters the hole at the optical center and impacts the image plane $Z = -f$ at the point $(-x, -f)$. Applying triangle similarities one easily concludes that $x/f = X/Z$. This suggests that a virtual image plane could be situated at $Z = f$, *i.e.* between the point and the optical center: the light ray would traverse it exactly at the point (x, f). This is handy because, while the image at the real plane is inverted, that in the virtual plane is not.

Extension to the 3D case is shown in Fig. 2.2, where just the virtual image plane is shown. Define the focal frame as $\mathcal{S}\{RDF\}$ with coordinates XYZ and the image frame as $\mathcal{I}\{RD\}$ with coordinates xy (see Chapter 1 for an explanation of the frame notations), and apply the pin-hole principle in the XZ and YZ planes. Obtain this way the pin-hole projection equation:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}, \tag{2.3}$$

where (x, y) are the coordinates of the projected point in the image plane.

Let us introduce now another mathematical construction that, together with the homogeneous coordinates[3], will allow us to write vision related equations in a more elegant form.

[2] Opaque surface with no specular light component: the incident light is reflected equally in all directions. The Lambertian assumption is very common in computer vision works like the one you are handling because, given scene lightning conditions, it insures a stable appearance of objects from disparate points of view.

[3] Defined in Section 1.2.4 of Chapter 1.

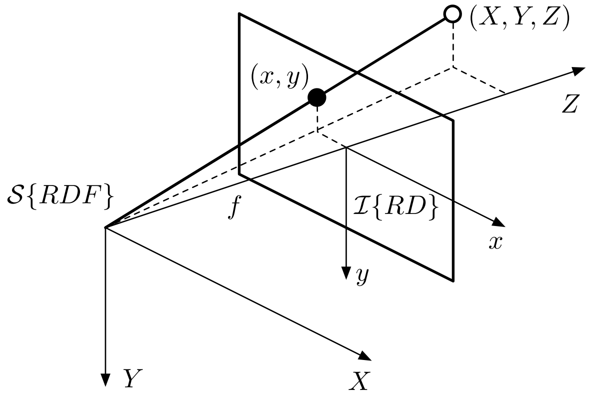

Figure 2.2: A convenient frame convention in the pin-hole camera. Adopting $\{RDF\}$ as the sensor frame together with $\{RD\}$ as the image plane frame leads to a highly comprehensive and intuitive, hence easy to remember, mathematical model.

Definition 2.1 (Projective space). Given a vector space V, we form the set of equivalence classes of non-zero elements in V under the relation of scalar proportionality. We consider vector $\mathbf{v}$ of being proportional to $\mathbf{w}$ if $\mathbf{v} = s\mathbf{w}$ for $s \neq 0 \in \mathbb{R}$. Further, we denote $\mathbf{v} \sim \mathbf{w}$ the fact that $\mathbf{v}$ is equivalent to $\mathbf{w}$.[4] □

See how, using homogeneous coordinates and projective spaces, relation (2.3) can be linearly rewritten as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

where the whole projective transformation is encoded in what we call the *projection matrix*:

$$\mathbf{P} \triangleq \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 4}. \tag{2.5}$$

This matrix is usually decomposed into

$$\mathbf{P} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where the two members may be defined as

$$\mathbf{K}_f \triangleq \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3} \qquad \mathbf{P}_0 \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 4}. \tag{2.6}$$

[4] Please note that mathematical definitions given are not absolutely rigorous; they are just enough for our purposes. For rigorous definitions visit for example `www.wikipedia.org`.

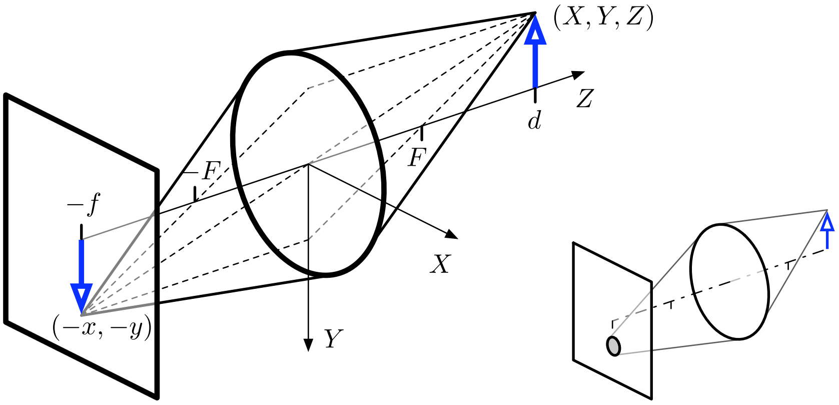

Figure 2.3: Ideal lens camera: focused and unfocused images.

If we denote in homogeneous coordinates $\underline{\mathbf{p}}^{\mathcal{S}} = [X, Y, Z, 1]^{\top}$ the 3D point in the sensor frame and $\underline{\mathbf{x}} = [x, y, 1]^{\top}$ the image point we can rewrite (2.4) as

$$s\,\underline{\mathbf{x}} = \mathbf{K}_f\,\mathbf{P}_0\,\underline{\mathbf{p}}^{\mathcal{S}}. \tag{2.7}$$

The matrix $\mathbf{P}_0$ is often referred to as the *normalized* (or '*standard*', or '*canonical*') projection matrix. It encodes the projection performed by the focal element into the image plane $Z = 1$. The matrix $\mathbf{K}_f$ encodes the scale factor derived from having the image plane at a distance $Z = f$ different to unity.

Remark 2.1 (Pin hole limitation). The key for taking good, sharp images with a pin-hole camera is that the hole must be *very small* so as to guarantee that the light coming from a point in the external world will impact within a very tiny region of the image plane. The limit for this perfection is an infinitely small hole, a situation that inevitably results in no light coming in. This means that little light is entering the camera, and hence that an unacceptable amount of time would be necessary to excite the sensing material in the image plane.[5] It is for this reason that you do not find pin-hole cameras in photography stores.[6] □

2.2.2 The thin lens camera

Fortunately enough, optics provides the *lens*, a device that can be used to our advantage. An ideal, convergent thin lens (Fig. 2.3) substitutes the tiny hole in the pin-hole camera by a circular *aperture* in the focal plane, centered at the optical center. Two points named *focus* are situated on the optical axis at a distance F at both sides of the aperture. Light rays emanating from one source point at one side of the lens are deflected, upon traversing the lens, and precisely converge into a single point, named *image*, at the other side. The position of this image point is determined by applying at least two among these three rules of elementary optics: *a)* Light rays reaching the focal plane in a direction parallel to the optical axis go out

[5] The total energy (in J) of light traversing a surface is an integral over time and over this surface of the light intensity (in W/m^2).

[6] You can always become an activist of using such cameras, like photographer Shi Guorui does. But you will have to construct them yourself.

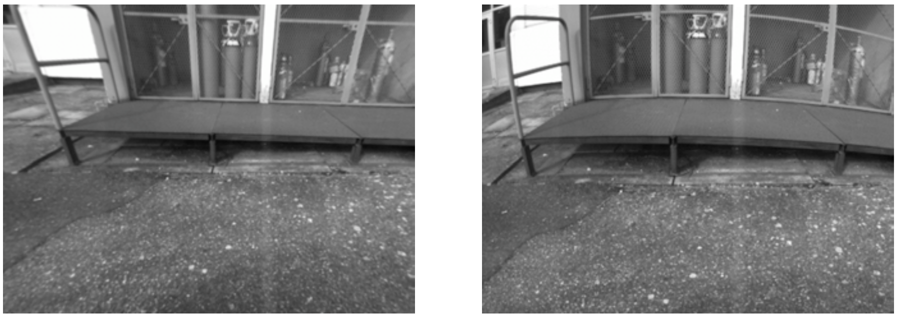

Figure 2.4: Typical effect of lens distortion on an image. Ideally projected image (*left*) and distorted image (*right*).

through the rear focus; *b)* Rays coming through the front focus go out parallel to the optical axis; and *c)* Rays passing through the optical center are undeflected.

To form the thin lens camera, an *image plane* parallel to the focal plane is situated precisely at the depth of this image point. This fixes the focal length f. Notice that rule *c)* above says that the pin-hole equations will hold for this camera. The advantage is a much greater amount of light reaching the image plane, permitting the sensing materials to get sufficiently excited, and thus allowing us to construct a practical, really working device.

Remark 2.2 (Lens limitation). The key for taking good, sharp images with a lens camera is that the image point must lie precisely on the image plane. Note that, while the focus F is a characteristic of the lens itself, the focal distance f is a characteristic of the camera, *i.e.* the couple {lens – image plane}. One can easily show that given a camera defined by f and F, only the source points in the plane $Z = d$, with d satisfying $1/F = 1/f + 1/d$, are properly focused. All other points closer and beyond this plane give place to unfocused images, blurred circles centered at the ideal, pin-hole image (Fig. 2.3 *right*). The size of these circles depends on the distance to the points and on the lens aperture's size: for R the aperture radius, r the blurred image radius and d the focused depth we have $\frac{r}{R} = \frac{f}{z} - \frac{f}{d}$. Adjusting the effective aperture radius R by means of a *diaphragm* allows us to control the depth range within which points will be imaged with a desired accuracy.[7] □

2.2.3 Distortion in the real lens camera

As expected, reality is more complex than theory. To construct real lenses, high quality glass is precisely machined. Big efforts are also devoted to maximize glass transparency, minimize diffraction or equalize color-dependent refraction: for many of the purposes of photography these are the most visible and unacceptable imperfections. But unlike most end-users of photographic cameras, we are concerned with obtaining mainly geometrical –not photometrical– information about the external world. It is for this reason that the phenomenon of *distortion* is crucial for us. This is what we examine now.

[7] Finally, to compensate for the variations in luminosity that playing with this aperture produces, the integration time or *shuttle speed* must be controlled accordingly. High gain, low noise photo-sensible devices should also be used to our advantage. These and other possible facts on photography are far beyond the interest of this work, but are crucial when choosing the right camera to use.

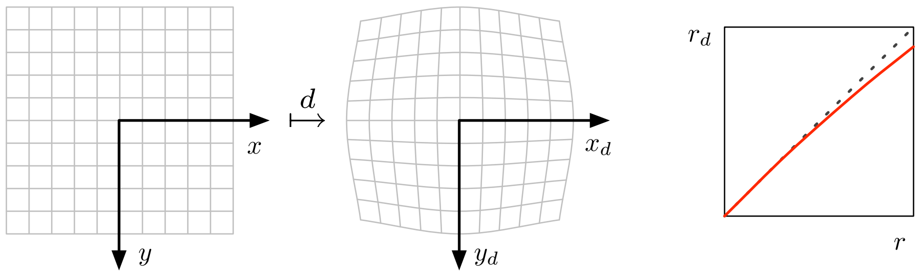

Figure 2.5: Mapping from ideal to distorted coordinates in a camera suffering from radial distortion (*left*). The polynomial describing the distorted-to-ideal radius ratio $r_d/r = (1 + d_2\, r^2 + d_4\, r^4 + \cdots)$ is very smooth and close to the identity (*right*).

Distortion (Fig. 2.4) is an aberration induced by the lens nature. It expresses the fact that the image point is no longer in a straight line with the optical center and the source point (hence violating the pin-hole principle). It has nothing to do with the focal distance f, and thus its effects are best observed and described in the normalized camera's image plane $Z = 1$. In this plane, distortion can be described generically by a bijective application d from the ideal coordinates to the distorted ones:

$$d :\ \mathbb{R}^2 \to \mathbb{R}^2; \quad (x, y) \mapsto (x_d, y_d) = d(x, y),$$

where the sub-index $(\cdot)_d$ denotes distorted coordinates.

Several distortion models $d(\cdot)$ have been studied by various authors in the last decade. A simple yet effective one arises if we assume perfectly round lenses precisely assembled parallel to the image plane, perfectly aligned perpendicular to the optical axis, that produce distortions only in the radial direction. The *radial distortion* (Fig. 2.5) is then defined by a scalar function $d(r)$ of the ideal image point radius $r = \sqrt{x^2 + y^2}$. It is usual for the ratio r_d/r to take a polynomial of even powers of r. This gives the radial distortion model

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = d\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = (1 + d_2\, r^2 + d_4\, r^4 + \cdots) \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.8}$$

which can be decomposed into[8]

$$r_d = d(r) = (1 + d_2\, r^2 + d_4\, r^4 + \cdots) \cdot r \tag{2.9}$$

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \frac{r_d}{r} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{2.10}$$

that is driven by the set of coefficients $\{d_2, d_4, \cdots\}$. Notice that the function $r_d = d(r)$ is a polynomial of odd powers of r. The length of the polynomial depends on the severity of the distortion: wide angle lenses can require up to three coefficients, while most of the times two or even only one will do. Tele-objective lenses may not require any distortion correction at all.

Several methods are available, sometimes in the form of software packages, to determine the polynomial coefficients of radial distortion via calibration procedures [Strobl et al. 2006;

[8] Notice that we are using the same name $d(\cdot)$ to mean both the vector function $\mathbf{x}_d = d(\mathbf{x})$ and the scalar one $r_d = d(r)$. This should not confuse the reader.

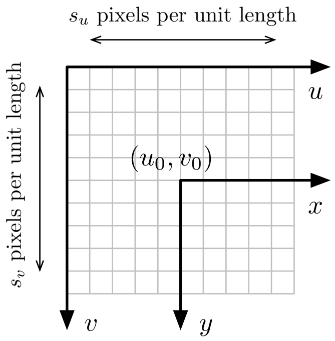

Figure 2.6: Transforming from metric to pixel coordinates.

Zhang 1999; Devy et al. 1997]. They are not covered in this work as they do not contribute to the comprehension of the phenomenon.

2.2.4 The image in pixel coordinates

Once on the image plane, the image must be transformed into some sort of information that permits us to extract it from the camera and provide it to the agent that will interpret it. When this agent is a computer, it is convenient to divide the image into a huge number of picture elements named *pixels*[9], distributed in a rectangular matrix form, which are supposed to contain uniform color and luminosity. The information of each pixel is encoded into digital words and transmitted to the computer at regular time intervals[10]. This way, an image sequence ready to be read by a computer consists of a table of digital words (a matrix of integer numbers) which is being updated at regular time intervals.

Let us concentrate in one of such images, that without loss of generality we will consider to be monochromatic. Before sampling, an image I may be re-viewed[11] as an application that associates the points of the plane with a positive number indicating its luminosity

$$I: \mathbb{R}^2 \to \mathbb{R}_+; \quad (u,v) \mapsto I(u,v) \tag{2.11}$$

where (u,v) are the coordinates in pixel units of the image plane.

Pixel coordinates (Fig. 2.6) usually start at one corner of the image plane. The matrix has s_u u-pixels and s_v v-pixels per unit length[12], and the optical axis intersects the image plane at the principal point (u_0, v_0). With a convenient $\{RD\}$ alignment of both coordinate systems we get the following mapping in homogeneous coordinates:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_u & 0 & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.12}$$

[9] This term is a contraction of the words *picture* and *element*.

[10] We see that three different kinds of sampling are taking place: *a) Spatial sampling* is performed by the image transducer, a planar matrix of tiny photo-sensible cells, the pixels, that transform the incident light energy into some kind of electric signal (voltage or current). *b)* Photometric *depth sampling* is performed on each pixel by analog-to-digital encoders which *c)* are fired at regular intervals by clock signals, thus performing *time sampling*.

[11] The image was already defined in (2.2).

[12] Pixels are reasonably rectangular, but not necessarily square.

If the pixels are not rectangular but parallelogram-shaped the matrix in the expression above takes a more general form. We name it $\mathbf{K}_s$ and define it by

$$\mathbf{K}_s \triangleq \begin{bmatrix} s_u & s_\theta & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.13}$$

Let us recall now expression (2.7). It relates the position of a 3D point to its ideal image. In the absence of distortion, it is clear that we can obtain the pixel coordinates $\underline{\mathbf{u}} = [u, v, 1]^\top$ with

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_u & s_\theta & u_0 \\ 0 & s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

which is shown here in its compacted form:

$$s\,\underline{\mathbf{u}} = \mathbf{K}_s\,\mathbf{K}_f\,\mathbf{P}_0\,\underline{\mathbf{p}}^{\mathcal{S}}. \tag{2.14}$$

The matrices $\mathbf{K}_f$ and $\mathbf{K}_s$ contain parameters that are *intrinsic* to a particular camera: the focal length and the pixels matrix parameters. Their product is another interesting matrix named the *intrinsic matrix*:

$$\mathbf{K} \triangleq \mathbf{K}_s\,\mathbf{K}_f = \begin{bmatrix} f \cdot s_u & f \cdot s_\theta & u_0 \\ 0 & f \cdot s_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & \alpha_\theta & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.15}$$

Its entries have the following geometrical interpretation:

- u_0, v_0: u- and v- coordinates of the principal point in pixels,
- α_u, α_v: the focal length in number of u- or v- pixels,
- $\alpha_\theta = f s_\theta$: skew of the pixel, often close to zero.

2.2.5 Summary of the perspective camera model

To summarize, let us write the whole sequence of operations to obtain the image pixel of a point in the 3D space. To further generalize the formulation, an additional step of 3D frame transformation (from frame $\mathcal{F}$ to sensor $\mathcal{S}$) is included by means of the homogeneous matrix $\mathbf{H}^{\mathcal{SF}}$ (see Remark 2.3 and Section 1.2.4 in Chapter 1). This matrix defines the pose of the sensor in an external reference frame and is therefore named the *extrinsic matrix*. Its entries (a translation vector and a rotation matrix) get the name of *extrinsic parameters*. The perspective camera model including extrinsic parameters is then, in the absence of distortion, the following linear expression in homogeneous coordinates:

$$s\,\underline{\mathbf{u}} = \mathbf{K}\,\mathbf{P}_0\,\mathbf{H}^{\mathcal{SF}}\,\underline{\mathbf{p}}^{\mathcal{F}}. \tag{2.16}$$

In the presence of lens distortion, the non-linear operator $d(\cdot)$ must be inserted between the projection and the transformation to pixel coordinates, that is, between $\mathbf{P}_0$ and $\mathbf{K}$. We proceed in sequential steps as indicated below (notice that equations (2.17) and (2.19) are in homogeneous coordinates while (2.18) is not):

$$s\,\underline{\mathbf{x}} = \mathbf{P}_0\,\mathbf{H}^{\mathcal{SF}}\,\underline{\mathbf{p}}^{\mathcal{F}} \tag{2.17}$$
$$\mathbf{x}_d = d(\mathbf{x}) \tag{2.18}$$
$$\underline{\mathbf{u}} = \mathbf{K}\,\underline{\mathbf{x}_d} \tag{2.19}$$

Finally, say that it will also be convenient to write the intervening elementary operations in function form. We have

$$\mathbf{p}^{\mathcal{S}} = toFrame(\mathcal{S}, \mathbf{p}^{\mathcal{F}})$$
$$\mathbf{x} = project(\mathbf{p}^{\mathcal{S}}) \triangleq \frac{1}{Z^{\mathcal{S}}}\begin{bmatrix} X^{\mathcal{S}} \\ Y^{\mathcal{S}} \end{bmatrix} \tag{2.20}$$
$$\mathbf{x}_d = distort(\mathbf{d}, \mathbf{x}) \triangleq d(\mathbf{d}, r)\cdot \mathbf{x} \tag{2.21}$$
$$\mathbf{u} = pixellize(\mathbf{k}, \mathbf{x}_d) \triangleq \begin{bmatrix} \alpha_u x_d + \alpha_\theta y_d + u_0 \\ \alpha_v y_d + v_0 \end{bmatrix} \tag{2.22}$$

where $toFrame(\cdot)$ has been defined in Chapter 1, $r = \sqrt{x^2 + y^2}$, $\mathbf{d} \triangleq [d_2, d_4, \cdots]^\top$ is a vector containing the distortion polynomial coefficients, and $\mathbf{k} \triangleq [u_0, v_0, \alpha_u, \alpha_v, \alpha_\theta]^\top$ is a vector containing the set of intrinsic camera parameters.

The complete perspective camera function can then be written for a point in the external frame $\mathcal{F}$ as

$$\mathbf{u} = \mathbf{h}(\mathcal{S}, \mathbf{k}, \mathbf{d}, \mathbf{p}^{\mathcal{F}}) \triangleq pixellize\Big(\mathbf{k}, distort\big(\mathbf{d}, project(toFrame(\mathcal{S}, \mathbf{p}^{\mathcal{F}}))\big)\Big). \tag{2.23}$$

Remark 2.3 (Camera-to-sensor frame transformation). Two differentiated frames are defined for a perspective camera (Fig. 2.7): the body or *camera frame* $\mathcal{C}\{FLU\}$ is used to easily specify the position of the object 'camera' in the robot (Chapter 1. For example, a camera looking forward gets the null Euler angles, or the identity rotation matrix). The lens or *sensor frame* $\mathcal{S}\{RDF\}$ is used to simplify the pin-hole equations so that they are easier to

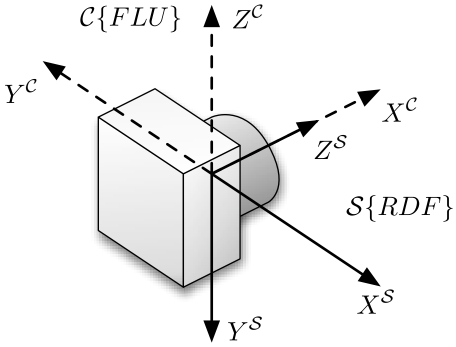

Figure 2.7: Camera body (*dashed*) and sensor (*solid*) frames. The associated rotation matrix (2.24) is easily determined from Proposition 1.1.

remember (recall Fig. 2.2). Both frames are related by a null translation and a constant rigid rotation (obtained by applying Proposition 1.1 on Fig. 2.7):

$$\mathbf{t}^{\mathcal{CS}} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{R}^{\mathcal{CS}} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{2.24}$$

This way, when the object camera pose $\mathcal{C}\{FLU\}$ is specified (for example via $\mathbf{H}^{\mathcal{FC}}$) the extrinsic matrix becomes (attention to super-indices!)

$$\mathbf{H}^{\mathcal{SF}} = \mathbf{H}^{\mathcal{SC}}\,\mathbf{H}^{\mathcal{CF}} = (\mathbf{H}^{\mathcal{CS}})^{-1}\,(\mathbf{H}^{\mathcal{FC}})^{-1}. \tag{2.25}$$

□

Remark 2.4 (Model inversion). Notice that *distort*(·), *pixellize*(·) and *toFrame*(·) are invertible applications, while *project*(·) is not. Hence, and because of projection, one will not be able to obtain, from a single image point, the full position of the 3D point that produced it: one of the three dimensions is lost. □

Remark 2.5 (Projection is great). The above remark may seem discouraging, even unfair. However, it is precisely the projection operation the one that permits to fit, inside a finite, small sized device such as an eye or a camera, all the immensity of the 3D world (See Fig. 2.8: you have your friend, the moon and some stars inside the same image!). Moreover, this projection will reveal great detail of nearby objects, while neglecting details of distant ones. This is also important: if we perceived all the details of all the external 3D space, how in the world could we separate the important from the vain? For any living creature, nearby things are more important for surviving; remote ones become important only if they are very big. This should also be true for a 'living machine' such as a robot. □

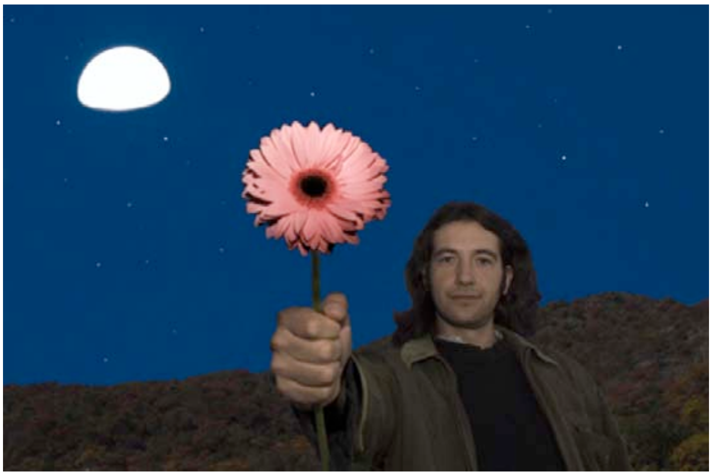

Figure 2.8: Projection is great! Photo courtesy of Xavier Soria and Max Pruden.

2.3 The perspective camera inverse model

As stated in Remark 2.4 one dimension is lost during projection. This section describes the inversion of the projection operation, that we will name *back-projection*. It can be defined up to an unknown scale factor which accounts for the lost dimension.

The inversion will be made by steps: *1)* the pixel unmapping, *2)* the distortion correction and *3)* the back-projection from the normalized camera. The first one is trivial. The second one is tricky because of the non-linear nature of the distortion. The third one is fundamental for the dimension loss as stated.

2.3.1 Inversion of the pixel mapping

Simply do

$$\underline{\mathbf{x}_d} = \mathbf{K}^{-1}\,\underline{\mathbf{u}} \tag{2.26}$$

with

$$\mathbf{K}^{-1} = \begin{bmatrix} 1/\alpha_u & -\alpha_\theta/\alpha_u\alpha_v & (\alpha_\theta v_0 - \alpha_v u_0)/\alpha_u\alpha_v \\ 0 & 1/\alpha_v & -v_0/\alpha_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.27}$$

In function form:

$$\mathbf{x}_d = depixellize(\mathbf{k}, \mathbf{u}) \triangleq \begin{bmatrix} (u-u_0)/\alpha_u - \alpha_\theta(v-v_0)/\alpha_u\alpha_v \\ (v-v_0)/\alpha_v \end{bmatrix}, \tag{2.28}$$

with $\mathbf{k} = [u_0, v_0, \alpha_u, \alpha_v, \alpha_\theta]^\top$ the vector of intrinsic parameters.

2.3.2 Distortion correction

The fact of distortion $d(\cdot)$ being bijective will allow us to recover the ideal image from the distorted one. We wish to invert $d(\cdot)$ and obtain the *correction* application $c(\cdot)$ from the distorted coordinates to the ideal ones:

$$c:\ \mathbb{R}^2 \to \mathbb{R}^2; \quad (x_d, y_d) \mapsto (x, y) = c(x_d, y_d) \triangleq d^{-1}(x_d, y_d). \tag{2.29}$$

Unfortunately, an analytical expression of the inverse of a polynomial is not easy to find, and if so it will have a complicated form. One method to avoid analytical inversion is to perform back-mapping. Let $I(x, y)$ be the ideal image and $I_d(x_d, y_d)$ be the distorted one. It is immediate to see that

$$I(x, y) = I_d(d(x, y)).$$

This operation is usually referred to as *distortion correction*, which is often tabulated to speed up calculations. It has a major disadvantage: as we do not know *a-priori* which pixel (x, y) corresponds to the measured (x_d, y_d), we are obliged to reconstruct the whole image. If this is what we want it is perfectly alright to do so.

However, if what we really want is to determine a small, sparse set of corrected pixels from the measured, distorted ones (to be able to find the 3D points that produced them via back-projection), we will economize a precious amount of computing resources if we only calculate the undistorted coordinates of the points we are interested in. This obliges us to invert the distortion function.

The analytical inversion of a polynomial is only feasible up to the fourth degree [Abel 1928]. This limits the inversion of radial distortion models to those of only one parameter.[13] For higher order models alternative solutions are needed. One method is to tabulate the function. Another one is to run, on-line, an iterative solver to find the roots r of the distortion equation 2.9 for any given r_d.

I find both solutions unsatisfactory. In fact, during the calibration procedure that determined the polynomial coefficients $\mathbf{d} = (d_2, d_4, \cdots)$, we could have determined also the optimal coefficients $\mathbf{c} = (c_2, c_4, \cdots)$ of its inverse function $c(\cdot)$, which could have been defined with a kernel of the same structure of the distortion one (which is itself arbitrary). Let us retain this last possibility: let us arbitrarily choose the kernel of a correction function $c(\cdot)$ so that the ratio r/r_d is another polynomial of even powers of r_d:

$$\mathbf{x} = c(\mathbf{x}_d) \triangleq (1 + c_2\, r_d^2 + c_4\, r_d^4 + \cdots)\, \mathbf{x}_d \tag{2.30}$$

which can be decomposed into

$$r = c(r_d) = (1 + c_2\, r_d^2 + c_4\, r_d^4 + \cdots) \cdot r_d \tag{2.31}$$

$$\mathbf{x} = \frac{r}{r_d}\mathbf{x}_d \tag{2.32}$$

with $r_d = \sqrt{x_d^2 + y_d^2}$. In function form we have

$$\mathbf{x} = correct(\mathbf{c}, \mathbf{x}_d) \triangleq (1 + c_2\, r_d^2 + c_4\, r_d^4 + \cdots)\, \mathbf{x}_d. \tag{2.33}$$

To my knowledge, and quite surprisingly, no solution has been proposed to include this correction as part of the calibration procedure, and the available software packages don't provide anything else than the direct distortion calibration.

We give a method to optimally determine the correction parameters. It starts assuming that a certain radial distortion model $r_d = d(r)$ is available, because it has been obtained with one of the existing camera calibration procedures. This calibration also provided the camera intrinsic parameters $\mathbf{k} = (u_0, v_0, \alpha_u, \alpha_v, \alpha_\theta)$. We define the correction operation as an approximation $c(r_d) \approx d^{-1}(r_d)$ in the least-squares sense. The goal is to obtain, given the distortion parameters $\mathbf{d} = (d_2, d_4, \cdots)$, the correction parameters $\mathbf{c} = (c_2, c_4, \cdots)$ that best approximate the equation (2.31) to the inverse of (2.9) —although any other distortion kernels could be used.

We proceed as follows: from the direct model $r_d = d(r)$ (2.9) we generate a data set of N corresponding pairs $\{r_i, r_{d,i}\}, i \in [1, N]$. This data set must be representative of the whole range of possible image radius. In the normalized image plane of a camera with intrinsic parameters $\{u_0, v_0, \alpha_u, \alpha_v, \alpha_\theta\}$, the maximal radius is well approximated with

$$r_{max} = \sqrt{\left(\frac{u_0}{\alpha_u}\right)^2 + \left(\frac{v_0}{\alpha_v}\right)^2} \tag{2.34}$$

so we simply take $r_i = i \cdot r_{max}/N$ and $r_{d,i} = d(r_i)$. Then we find the set of parameters $\mathbf{c}$ that minimizes the error function

$$f(\mathbf{c}) = \sum_{i=1}^{N} \left[r_i - c(r_{d,i})\right]^2. \tag{2.35}$$

[13] Recall from Section 2.2.3 that the distortion function $r_d = d(r)$ is a polynomial of odd powers of r.

In the cases where $c(\cdot)$ is linear in the components of $\mathbf{c}$, as it is for the radial distortion model, (2.35) corresponds to a linear least-squares optimization problem. It is solved by means, for instance, of the pseudo-inverse method by writing the equations $r_i = (1 + c_2\, r_{d,i}^2 + c_4\, r_{d,i}^4 + \cdots)\, r_{d,i}$ for $1 \leq i \leq N$ as the linear system

$$\begin{bmatrix} (r_{d,1})^3 & (r_{d,1})^5 & \cdots \\ \vdots & & \\ (r_{d,N})^3 & (r_{d,N})^5 & \cdots \end{bmatrix} \begin{bmatrix} c_2 \\ c_4 \\ \vdots \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_N \end{bmatrix} - \begin{bmatrix} r_{d,1} \\ \vdots \\ r_{d,N} \end{bmatrix}$$

which, re-written as $\mathbf{R}_d\,\mathbf{c} = (\mathbf{r} - \mathbf{r}_d)$, leads to the least-squares optimal solution

$$\mathbf{c} = \left[(\mathbf{R}_d^\top\, \mathbf{R}_d)^{-1} \mathbf{R}_d^\top \right] (\mathbf{r} - \mathbf{r}_d). \tag{2.36}$$

Obviously, the error in the correction function has to be added to the error existing in the distortion function used as reference. This would not have happened if both functions had been simultaneously determined during the calibration process. For this reason, we may try to make the error of this stage significantly smaller than that of the distortion. Should this be necessary, we can adjust the error by choosing the correction kernel polynomial to be longer (of higher degree, thus more accurate) than the distortion one.

We illustrate this method with an example directly taken from our experiments.

Example 2.1 (Inversion of the distortion function):
The Matlab calibration toolbox was used to calibrate a camera. We obtained the intrinsic and distortion parameters summarized in Table 2.1.

Table 2.1: Calibration data.

u_0	v_0	α_u	α_v	α_θ	d_2	d_4
516.686	355.129	991.852	995.269	0.0	-0.301701	0.0963189

From (2.34) the maximum radius in the image plane is $r_{max} = 0.6314$. A number of $N = 100$ uniformly spaced data pairs $\{r_i, r_{di}\}$ is sampled from the function $r_d = d(r) = (1 + d_2\, r^2 + d_4\, r^4) \cdot r$ between $r = 0$ and $r = r_{max}$. A linear least-squares optimization via pseudo-inverse is performed to minimize (2.35) with the correction function kernel $r = c(r_d) = (1 + c_2\, r_d^2 + c_4\, r_d^4) \cdot r_d$, of equal length to that of the distortion kernel. The obtained optimal set of correction parameters is

$$c_2 = 0.297923 \qquad c_4 = 0.216263.$$

The error vector is defined $e_i = r_i - c(r_{di})$. Its maximum and standard deviation values are:

$$\begin{aligned} e_{max} &\approx 4 \cdot 10^{-5} \\ \sigma_e &\approx 1.75 \cdot 10^{-5}. \end{aligned}$$

These values can be translated into horizontal pixel units by simply multiplying by α_u:

$$\begin{aligned} e_{max} &\approx 0.04\text{pix} \\ \sigma_e &\approx 0.0175\text{pix}. \end{aligned}$$

These results are plotted in Fig. 2.9. We see that in this case the correction pixel error is very small and it is not necessary to make the correction kernel longer than the distortion one. ♣

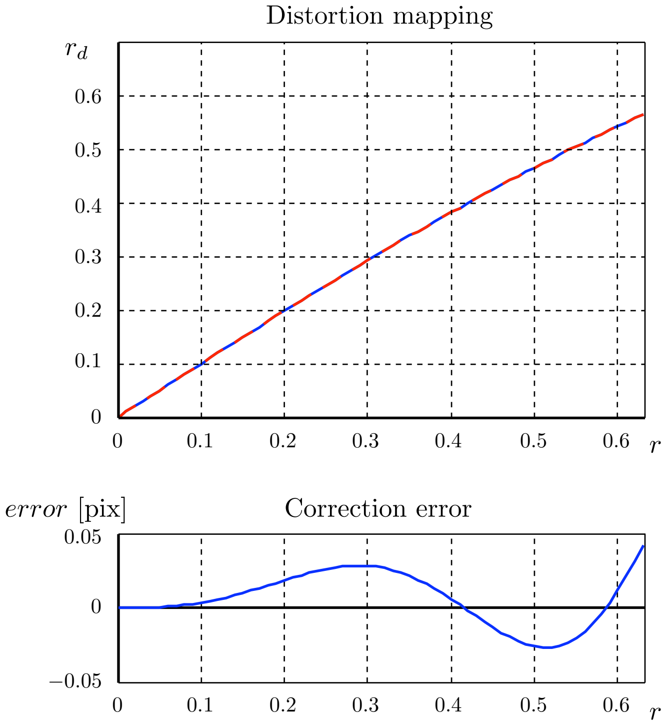

Figure 2.9: The least-squares approximation of the correction function. Distortion $r_d = d(r)$ and correction $r = c(r_d)$ mappings show a very good curve fit (*top*). The correction function error with respect to the distortion one (*bottom*) shows very small maximum errors which are attained at the corners of the image, where the radius is bigger.

2.3.3 Back-projection from the normalized camera

From the projection equation (2.3) with $f = 1$ and a point in the image plane (x, y), the equation of the line that starts at the optical center and passes through the image point (see also Fig. 2.10) is written in parametric form:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.37}$$

The third row of (2.37) shows that the line's parameter s is precisely the distance from the 3D point to the focal plane $Z = 0$. We name this distance the *depth* of the point. To refer to the lost dimension in a more concise way it is common to say that '*the depth of a 3D point is not observable from a single picture shot*'.

In function form we have

$$\mathbf{p}^{\mathcal{S}} = \mathit{backProject}(s, \mathbf{x}) \triangleq s \begin{bmatrix} x & y & 1 \end{bmatrix}^\top \tag{2.38}$$

2.3.4 Summary of the perspective camera inverse model

In the absence of lens distortion, the inverse model in the sensor frame is simply

$$\mathbf{p}^{\mathcal{S}} = s\,\mathbf{K}^{-1}\,\underline{\mathbf{u}}, \tag{2.39}$$

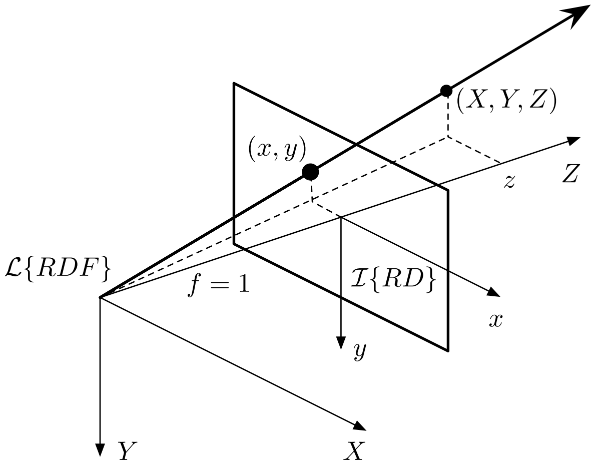

Figure 2.10: The back-projection. All points (X,Y,Z) in the back-projected ray of the image point (x,y) satisfy $[X,Y,Z]^\top = s\,[x,y,1]^\top$ with $s \in [0,\infty)$.

where s is an undetermined depth. With lens distortion the model has to be split into three stages

$$\begin{aligned}\underline{\mathbf{x}_d} &= \mathbf{K}^{-1}\underline{\mathbf{u}} \\ \mathbf{x} &= c(\mathbf{x}_d) \\ \mathbf{p}^{\mathcal{S}} &= s\,\underline{\mathbf{x}}.\end{aligned}$$

In function form, the complete perspective camera inverse function $\mathbf{g}(\cdot)$ is defined by

$$\begin{aligned}\mathbf{p}^{\mathcal{S}} &= \mathbf{g}(s,\mathbf{c},\mathbf{k},\mathbf{u}) \\ &\triangleq \mathit{backProject}\Big(s, \mathit{correct}\big(\mathbf{c}, \mathit{unpixellize}(\mathbf{k},\mathbf{u})\big)\Big).\end{aligned} \tag{2.40}$$

2.4 Basic feature-based image processing

Making a machine understand a moving image remains an open problem. Doing it in real-time, while the robot is moving and needs to act reactively, becomes almost impossible. A sentence like “Follow that car”, so common in thriller films where the addressee is a taxi driver, becomes absolutely meaningless for a robot unless it knows what cars look like, how to identify them in the image, and how to situate them in 3D space, all these questions being resolved quickly enough to permit reaction. If the car is moving, the robot will additionally need to estimate its speed and predict its maneuvers.

This is all what this thesis is about, but is by no means all what this thesis can solve. We want to focus in the real time solution of a part of this vast problem. For this, we will leave behind those tasks with higher level of abstraction such as object recognition and classification, or behavior prediction. As a consequence we will be exploiting a very little

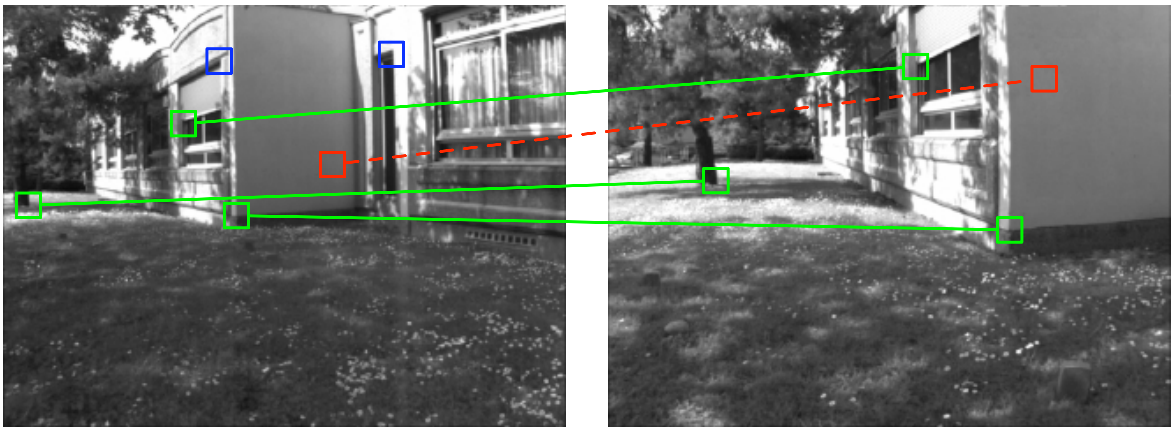

Figure 2.11: Feature detection and matching. Features are salient points; they posses some kind of uniqueness that allow the machine to identify them in other images by evaluating a similarity measure (*green*). Without this uniqueness, a good similarity would lead to matched features that do not correspond to the same 3D point (*red*). Some features may not find their corresponding matches in other images (*blue*).

part of the information an image can provide: we will seek for procedures able to recover, online and incrementally, some geometrical properties of the surrounding world. Thus only tractable geometrical information from the images will be explicitly exploited. This resumes in our case to the position of certain 'features' in 2D images, which we will uniquely associate to particular 'landmarks' in the 3D space. Photometrical information will only be used to perform this association via 'feature matching'.

This is actually feasible thanks to the possibility of reusing a precious material set up by the computer vision community in the last few years: without really 'understanding' much on the images, a computer can easily and rapidly perform the following tasks (Fig. 2.11):

1. **Identify salient points.** Salient points are those that are locally distinguishable, that possess some strong particularity that make them 'unique' in some sense. These points are usually called *features*, and the process of identifying them in an image is normally referred to as *feature detection*. In gray-level images a feature may correspond to a point where the luminosity variation in all directions is locally maximal. This analysis of the image derivatives is at the heart of most of the best known feature detectors.

2. **Match salient points in different images.** The task of finding the same point in another image is usually called *feature matching*. From the above statements, the uniqueness of a feature is determined by its close neighborhood. If we memorize this neighborhood as the feature's signature, for example in the form of a small rectangular patch like those in Fig. 2.11, we will be able to find it in other images by simply scanning the new image for a similar enough signature.

That is, without neither detecting nor recognizing objects in the images, a robot is capable to select interesting points in its surrounding 3D world and track them for a certain number of images. In later chapters in this thesis we will see how to use this power to localize these points in the 3D space, building with them a sort of 'map', while simultaneously using every new observation to get self-localized in this map. By doing that, the robot fulfills a complete

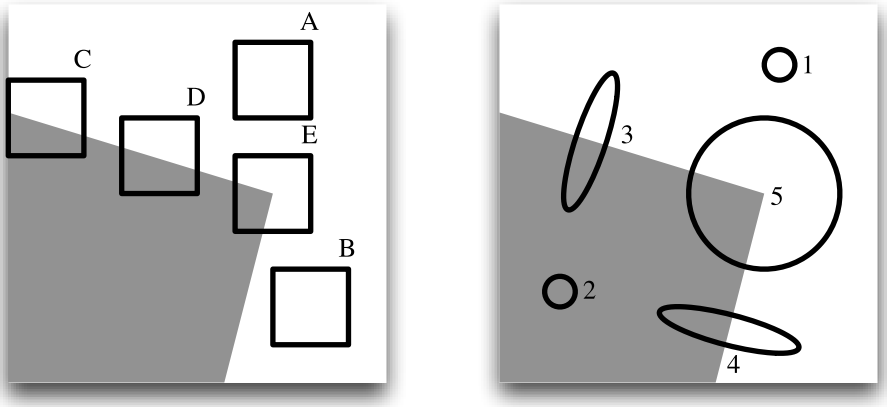

Figure 2.12: Edges and corners. *Left:* Interesting features to detect are those which will be easily and uniquely matched. *A, B:* Features lying on low-textured regions have poor uniqueness that lead to wrong matching; *C, D:* Features lying on edges present an indetermination along the edge line. They define line features. *E:* The only 'unique' point feature in this sample is at the corner. *Right:* Ellipses representing the rate of change of luminosity in a grey-level image. *1, 2:* Non-salient points have low rate of change; *3, 4:* Edges have high rate of change in only one direction; *5:* Corners have high rate of change in all directions.

task of true exploration. In the presence of moving objects in the scene, the robot will try to detect the moving points and determine their positions and velocities. Later and out of this work, we could define an object by grouping neighbor points, and a moving object by grouping neighbor points that have also very similar velocities.

This section is dedicated to present the rudiments of the basic techniques for feature detection and matching that we use in our works.

2.4.1 The Harris corner detector and some variations

To identify a salient point in a grayscale image the idea is to select those points where the gray level spacial derivatives are locally maximal (Fig. 2.12). We speak of *corner* if this rate of change is maximum in all directions. We may also speak of *edge* when this rate of change is important only in one direction.

The first derivatives $\mathbf{I}_u$ and $\mathbf{I}_v$ of the gray level of an image $\mathbf{I}(u, v)$ with respect to u and v, the horizontal and vertical coordinates respectively, may be accurately approximated by correlation with the so called interpolation and derivative masks [Farid and Simoncelli 1997], defined in general as the symmetric and anti-symmetric vectors

$$\begin{aligned}\mathbf{i}_n &= \begin{bmatrix} i_n & \cdots & i_1 & i_0 & i_1 & \cdots & i_n \end{bmatrix}^\top \\ \mathbf{d}_n &= \begin{bmatrix} -d_n & \cdots & -d_1 & 0 & d_1 & \cdots & d_n \end{bmatrix}^\top\end{aligned}$$

of length $2n+1$. The masks entries i_i and d_i are determined as a function of n by sampling a Gaussian and a Gaussian derivative. They allow us to adequately approximate the true image derivatives with the correlations

$$\mathbf{I}_u(u, v) = \mathbf{I}(u, v) \otimes \mathbf{d}_n^\top \otimes \mathbf{i}_n \tag{2.41}$$

$$\mathbf{I}_v(u, v) = \mathbf{I}(u, v) \otimes \mathbf{d}_n \otimes \mathbf{i}_n^\top \tag{2.42}$$

where $\otimes$ indicates the correlation operator. Sometimes, specially when speed is a must, we may simply use the so called Sobel masks $\mathbf{i}_1 = [1\ 2\ 1]^\top/4$ and $\mathbf{d}_1 = [-1\ 0\ 1]^\top/3$ or even their non-normalized version $i'_0 = 1$ and $\mathbf{d}'_1 = [-1\ 0\ 1]^\top$ which leads to

$$\mathbf{I}_u(u,v) = \mathbf{I}(u+1,v) - \mathbf{I}(u-1,v) \tag{2.43}$$

$$\mathbf{I}_v(u,v) = \mathbf{I}(u,v+1) - \mathbf{I}(u,v-1). \tag{2.44}$$

The Harris edge and corner detector [Harris and Stephens 1988] computes the derivatives of image $\mathbf{I}$ at every pixel (u,v) and builds with them the symmetric matrix

$$\mathbf{M}(u,v) = \mathbf{g}_m(\sigma) \otimes \begin{bmatrix} \mathbf{I}_u^2 & \mathbf{I}_u\mathbf{I}_v \\ \mathbf{I}_u\mathbf{I}_v & \mathbf{I}_v^2 \end{bmatrix} \tag{2.45}$$

where $\mathbf{g}_m(\sigma)$ is a 2D Gaussian mask of size $m \times m$ and variance σ^2 that acts as a weighting, smoothing correlator. The matrix $\mathbf{M}(u,v)$, when represented as an ellipse (see Appendix B), presents a major and a minor principal axes that correspond to the directions and strengths of the gray level maximum and minimum rates of change. The relation of these rates of change with the uniqueness of an image feature is also illustrated in Fig. 2.12. The matrix $\mathbf{M}$ conveniently encodes all the information of the amount of 'cornerness' or 'edgeness' a pixel has.

From the matrix $\mathbf{M}$, the Harris detector defines the 'cornerness' measure at every pixel as follows

$$H(u,v) = \det(\mathbf{M}) - k \cdot (\operatorname{trace}(\mathbf{M}))^2 \tag{2.46}$$

with $k = 0.04$. Local maxima of $H(u,v)$ that are also greater than a certain threshold correspond to corners in the image that will be used as features.

The above measure is discussed in some works as being unjustifiably arbitrary, notably because of the presence of parameter k. This alternative measure is used by Noble [1989]:

$$N(u,v) = \det(\mathbf{M}) / \operatorname{trace}(\mathbf{M}). \tag{2.47}$$

Myself, I am more confident with the criterion used by Shi and Tomasi [1994] which considers the 'cornerness' measure as the ellipse's minor semi-axis, which we recall it corresponds to the smallest eigenvalue of $\mathbf{M}$:

$$S(u,v) = m_{uu} + m_{vv} - \sqrt{(m_{uu} - m_{vv})^2 + 4\, m_{uv}^2} \tag{2.48}$$

with m_{ij} the entries of $\mathbf{M}$. Using this measure, a corner is that point where the smallest rate of change is locally maximal.

We notice that it is possible to choose the weighting mask in $\mathbf{M}$ to be non-Gaussian. By taking a uniform square mask, the Shi and Tomasi measure $S(u,v)$ optimizes the selectivity of correlation-based matchings with patches of the same size of the mask. That is, in order to optimize matching, the weighting mask is chosen to be uniform and of the same size as the feature patch descriptor. While this is claimed as an advantage in [Shi and Tomasi 1994], we have noticed that the resulting features are not well centered at the corner and thus they cannot define true 3D positions when a significant amount of scale change is present (Fig. 2.13).

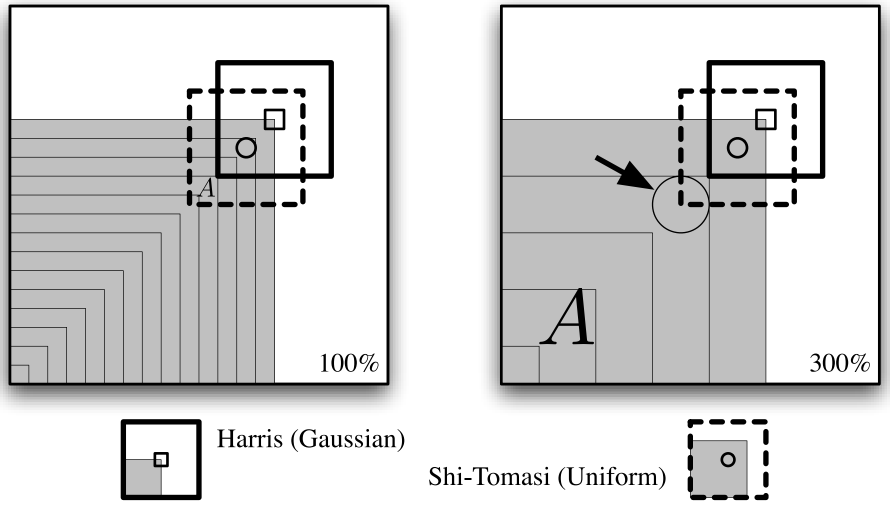

Figure 2.13: Gaussian versus uniform weighting masks in the presence of important scale change. By favoring information closer to the center, Gaussian masks give better corner-centered features that adequately resist scale changes. Uniform masks give 'inside-the-corner' features. When zoomed in, posterior matches point towards unstable 3D locations (the arrowed circle on the zoomed image is where the original patch, defined on the unzoomed image, was pointing).

2.4.2 Correlation-based feature matching

Feature matching may be performed by exploiting different principles (geometric based, appearance based, object-recognition based, and maybe others). We concentrate on the appearance based ones, as they are robust yet easy to define and fast to compute.

The feature's appearance is described by a medium-sized rectangular patch in the vicinity of the corner pixel detected by the feature detector above (Fig. 2.14). We call this patch the *reference patch*. In subsequent images, every pixel is assigned a patch of the same size that the reference one. The pixel that originated a particular patch is named the *base pixel* of the patch, and is normally chosen to be the central pixel.

A similarity measure may be assigned to each pixel in the new image. This measure is computed from the appearances of the reference patch and the pixel's associated patch, which

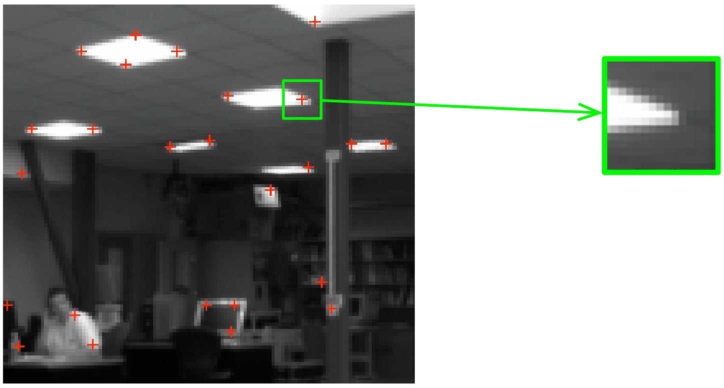

Figure 2.14: Feature detection (*red*) and patch definition (*green*).

must be obviously of the same size. By extremizing this measure we can identify the most similar patch and therefore the pixel that best 'matches' the reference one.

Different appearance-based similarity measures are presented in Table 2.2, where

- W is the window defining the patches, *i.e.* a rectangular region containing a certain number of pixels and centered at the base pixel.
- N is the number of pixels in the window W.
- I is the reference patch defined at the time of feature detection.
- J is the patch in the current image defined around the pixel we are testing.
- $\sum_W I$ is the sum of the elements of I inside the window W, *i.e.*

  $$\sum_W I \triangleq \sum_{(u,v)\in W} I(u,v).$$

  This is valid for all sums $\sum_W(\cdot)$, for instance

  $$\sum_W IJ = \sum_{(u,v)\in W} I(u,v)J(u,v).$$

- $\bar{I}$ and $\bar{J}$ are the means of the patches luminosities, *i.e.*

  $$\bar{I} = \frac{1}{N}(\sum_W I)$$

  and akin for $\bar{J}$.

- σ_I and σ_J are the standard deviations of the patches luminosities, *i.e.*

  $$\sigma_I = \sqrt{\frac{1}{N}\sum_W (I-\bar{I})^2}$$

  and akin for σ_J.

- C_I and C_J are the Census patches. For example, C_I is a binary patch where every entry is coded '1' if the luminosity of its corresponding pixel in I is greater than the base pixel, and '0' otherwise.

The characteristics of the similarity measures shown can be resumed as follows:

- SAD and SSD correspond to distance measures. At first sight, minimizing SSD will lead to a least-squares solution of the feature matching problem. This minimization is numerically performed by scanning over the image. SAD constitutes a faster computation alternative.
- Correlation coefficients CC, NCC, ZCC and ZNCC make use of the correlation operator used in signal processing also to quantify similarity. From CC, the zero-mean modification ZCC allows for invariant measurements with respect to brightness (additive white light). Using normalization of the patches luminosity with respect to their variance (NCC) leads to a measure which is invariant to luminosity contrast variations. The ZNCC measure is invariant to both brightness and contrast, thus to a wide range of luminosity changes.

Table 2.2: A collection of patch similarity measures, compared.

Similarity Measure	Acronym		Expression	Extremization
Sum of Absolute Differences	SAD	$=$	$\sum_W \lvert I - J \rvert$	min
Sum of Squared Differences	SSD	$=$	$\sum_W (I - J)^2$	min
Cross-Correlation	CC	$=$	$\sum_W I\,J$	max
Zero-mean CC	ZCC	$=$	$\sum_W (I - \bar{I})(J - \bar{J})$	max
Normalized CC	NCC	$=$	$\frac{1}{N} \sum_W \frac{I\,J}{\sigma_I\,\sigma_J}$	max
Zero-mean, Normalized CC	ZNCC	$=$	$\frac{1}{N} \sum_W \frac{(I-\bar{I})(J-\bar{J})}{\sigma_I\,\sigma_J}$	max
Census	Census	$=$	$\sum_W \mathrm{XOR}(C_I, C_J)$	max

- Finally, Census is a non-metric measurement which uses exclusively binary operators. It is specially interesting for ultra-fast computation in FPGA-based processing.

The normalization and centering of the ZNCC makes it a very robust measure over long sequences where lighting conditions are changing —notice that a moving object entering a shadowed area will significantly change its appearance, and that our measurement should not be affected. Additionally and very important for us, the fact of ZNCC being normalized allows us to define a similarity 'score' which can be used to evaluate the match quality or for match validation via thresholding. We give an alternative expression of the ZNCC which is more computationally efficient:

$$\mathrm{ZNCC} = \frac{(N \cdot S_{IJ} - S_I \cdot S_J)}{\sqrt{(N \cdot S_{II} - S_I^2)\,(N \cdot S_{JJ} - S_J^2)}} \tag{2.49}$$

where

$$S_I = \sum_W I; \quad S_J = \sum_W J; \quad S_{II} = \sum_W I^2; \quad S_{JJ} = \sum_W J^2; \quad S_{IJ} = \sum_W IJ.$$

Notice that S_I and S_{II} need to be computed only once for the whole lifetime of the landmark described by patch I, that is to say they can be stored as part of the landmark descriptor.

We show in Fig. 2.15 the ZNCC measure between a patch of 15 × 15 pixels and the patches associated to all its neighbor pixels in the horizontal direction. See how ZNCC $= 1$ for the exact match and ZNCC < 1 in all other cases. Observe also the measure's peak shape as a function of the original patch. Only the patch in the central figure was defined by a corner detector. Notice that we seek for sufficiently discriminating peaks that are also robust to small appearance variations.

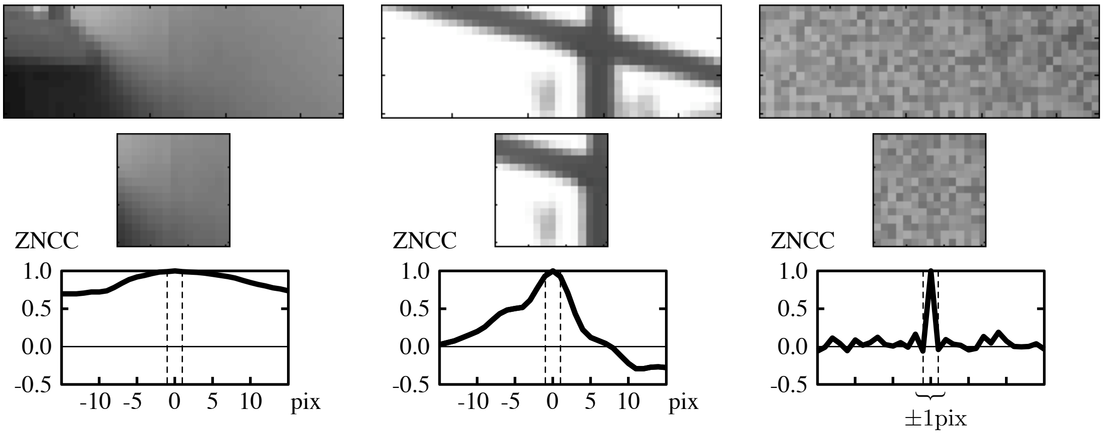

Figure 2.15: Image autocorrelation using the Zero-mean Normalized Cross-Correlation (ZNCC). A 15 × 15 pixels patch is correlated with its horizontal vicinity. *Left:* Poor texture leads to high autocorrelations in the pixel vicinity and hence bad discrimination. *Center:* Good features present good discrimination and a sufficiently wide correlation peak that allow for correct matching with reasonable appearance variations. *Right:* Too much texture (white noise) gives delta-shaped autocorrelations which prevent satisfactory matching when slight appearance changes are present.

Chapter 3

Filtering

> I think it was Laplace who wrote something like "Probability is the mathematical formulation of common sense".[1] I definitely agree. With the theory of probabilities men also reinvented knowledge, which is a prior to reasoning. And this, I find amazing.

3.1 Introduction

In a way, filtering is some kind of reasoning that takes previous knowledge and observed evidence together to infer the characteristics of a phenomenon of interest. This reasoning is basically reproducing the rules of common sense. To see it, consider this story:

> Somebody enters a room and closes the door afterwards. You are pretty sure: if you affirm that "he is inside the room, now" there is nothing to object here, it is rigorous common sense. But notice that you cannot actually see him inside. Now consider that, just before he entered, you heard a telephone ringing inside the room: now you are in position to add that, with a certain probability, "he is having a telephone conversation", although you cannot verify it. And this goes on as everyday life: if you know him, and if you were actually expecting a telephone call to arrive, you may wait until he comes out and ask "So?". And if at this point he answers "Tomorrow; ten o'clock" you may suddenly feel excited.

The great thing of all this is that everything that you have perceived has just helped to guide the way your knowledge on the situation was getting more and more precise. And that without your previous knowledge and the predictions you can make (you actually 'knew' that he was on the phone and what he was talking about), the information contained in 'he enters the door', 'a telephone rings', 'he comes out' and 'tomorrow, ten o'clock' is disconnected from reality and is therefore absolutely meaningless.

[1] Pierre-Simon de Laplace (1749-1827) wrote in French what in English reads "Probability theory is nothing but common sense reduced to calculation" and the longer one "The theory of probabilities is at bottom nothing but common sense reduced to calculus; it enables us to appreciate with exactness that which accurate minds feel with a sort of instinct for which ofttimes they are unable to account."

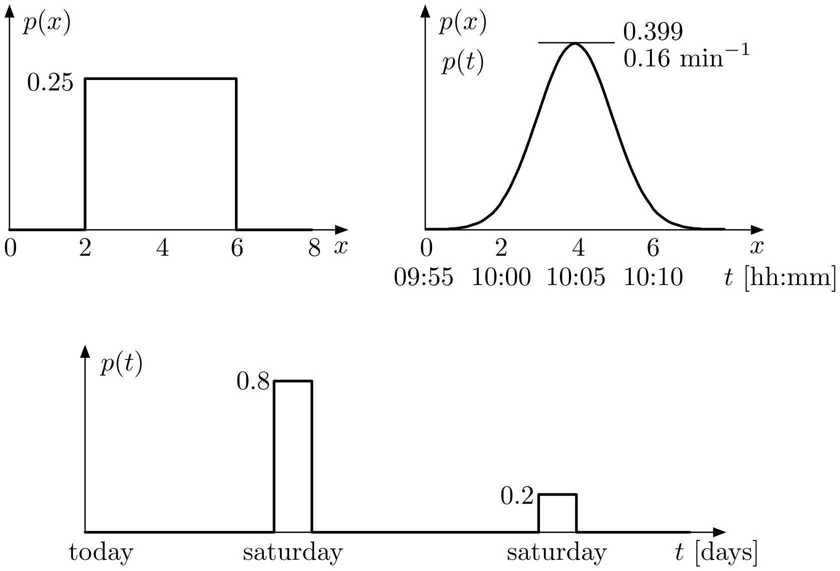

Figure 3.1: Three examples of *pdf* encoding different beliefs. *Top left*: Uniform distribution, 'X can be anything between 2 and 6' or simply '$2 \leq X \leq 6$'. *Top right*: Gaussian distributions, $p(x)$ says 'X is close to 4 with an uncertainty of ± 1'; $p(t)$ describes the knowledge 'Tomorrow, ten o'clock', something that is not likely to happen before 10:00, nor after 10:10, and expected around 10:05. *Bottom*: A bimodal distribution, 'I think it's this Saturday, but it could be next one.'

3.1.1 Some brief notions from the Theory of Probabilities

In the Theory of Probabilities the knowledge we have about the values that a random variable is likely to take[2] (what can be called the *belief*) is specified via a probability measure which can be represented in major cases by *probability density functions* (*pdf*), sometimes simply referred to as '*densities*' or '*distributions*'.[3] Informally, a probability density function $p_X(x)$ is the density of probability that a realization of the random variable X takes the value x. In the real scalar case $X \in \mathbb{R}$ this *pdf* definition is written as follows

$$p_X(x) \triangleq \lim_{dx \to 0} \frac{P(x \leq X < x + dx)}{dx} \tag{3.1}$$

where $P(A) \in [0, 1]$ is the probability measure of the set A, or more informally, the probability of A being true. We have illustrated some examples of *pdf* in Fig. 3.1. A *pdf* $p_X(x)$ has the following properties

$$p_X(x) \geq 0 \quad \forall x, X \in \mathbb{R}$$
$$\int_{-\infty}^{\infty} p_X(x) dx = 1.$$

In the notation, the subscript denoting the realization is often ignored and $p_X(x)$ is simply written $p(x)$.

[2] In mathematics 'things' or 'situations' are conveniently encoded into functions or variables that take values within a certain set of possibilities.

[3] Formally, it is always possible to extend the notion of *pdf* to all probability measures by considering *general functions* (L. Schwartz's distribution).

Another important function in probabilities is the *Expectation operator* $\mathbb{E}[\cdot]$ defined as the integral of a function with respect to the the probability measure as follows

$$\mathbb{E}\big[f(x)\big] \triangleq \int_{-\infty}^{\infty} f(x)\,p(x)\,dx. \tag{3.2}$$

The expectation operator is linear, *i.e.* it satisfies $\mathbb{E}[kf(x)] = k\mathbb{E}[f(x)]$ and $\mathbb{E}[f(x)+g(x)] = \mathbb{E}[f(x)] + \mathbb{E}[g(x)]$. We further define the *moments* of order n of X as

$$m_n \triangleq \mathbb{E}\big[x^n\big] \tag{3.3}$$

which permit to identify the *mean* or *expected value* of the variable X with the first moment

$$\bar{x} \triangleq \mathbb{E}[x] \tag{3.4}$$

and the *variance* of X with the second *central moment*[4]

$$\mathbb{V}[x] \triangleq \mathbb{E}[(x-\bar{x})^2] = \mathbb{E}[x^2] - \big(\mathbb{E}[x]\big)^2. \tag{3.5}$$

We conclude with the *standard deviation* of X which is defined by

$$\sigma_x \triangleq \sqrt{\mathbb{V}[x]} = \sqrt{\mathbb{E}[(x-\bar{x})^2]}. \tag{3.6}$$

Plenty of other facts about Probabilities are not included here as the main purpose of this presentation is to give a comprehensible overview of the filtering problem and its solutions. The reader should however be familiar with them in order to correctly follow this chapter. Of special importance are the multivariable extension of the definitions just given, the conditional probability, the Bayes rule, the notion of independence of random variables, the Markovian processes and the marginal densities. These concepts will appear in the next sections without previous notice nor proof, and often without even being mentioned: they will be just implicit in the exposed material.[5]

3.1.2 The filtering problem

Filtering is refining the knowledge we have about the state of a system from the information provided by the measurements we make on it. The theory of probabilities will permit us to solve this problem automatically, and therefore to build a reasoning machine.

Formulation of the filtering problem

Consider a dynamic system Σ which is affected by some random perturbations. Consider a set of sensors providing noisy information on this system at a more or less regular basis. Consider also that, as engineers, we have some knowledge on the way this system evolves with time, and knowledge on how the measurements relate to the current state of the system.[6] Further consider that we know which random character the system perturbations have and how they affect the system's dynamics, and akin for the measurements noise. Finally consider that we have some information about the initial state of the system.

[4] The notion of central moment should be obvious after inspection of definition (3.5).

[5] I tried to write in *italic* form every new concept appearing in the text for the first time.

[6] This means in our case that we correctly understood Chapters 1 and 2.

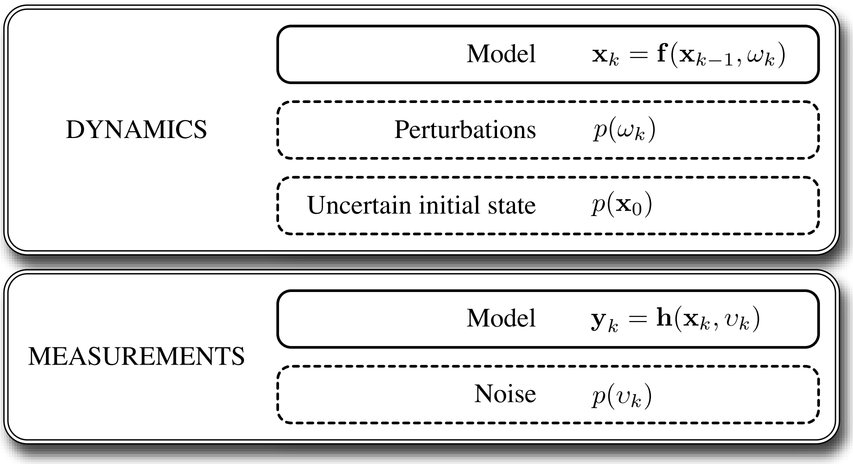
DYNAMICS
Model $\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \omega_k)$
Perturbations $p(\omega_k)$
Uncertain initial state $p(\mathbf{x}_0)$
MEASUREMENTS
Model $\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \upsilon_k)$
Noise $p(\upsilon_k)$

Figure 3.2: Solving the filtering problem requires writing down into mathematical expressions all the knowledge available about the system, both deterministic (or certain, in *solid line*) and stochastic (or uncertain, in *dashed line*).

All this knowledge on our system is summarized in Fig. 3.2. It specifies both deterministic and stochastic knowledge via mathematical expressions, which we revise now. The deterministic equation

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \omega_k) \tag{3.7}$$

is called the *evolution equation* and expresses in a discrete-time, state-space formalism the *Markovian* evolution of the system state $\mathbf{x}_k$ from time instant $k-1$ to k when it is subject to an uncertain control action[7] ω_k. The second deterministic equation

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \upsilon_k) \tag{3.8}$$

is called the *measurement equation* and describes the noisy measurements the sensors will provide at time k.

The stochastic knowledge is specified via probability density functions. This way, the *pdf*

$$p(\omega_k) \tag{3.9}$$

gets the name of *process noise* and expresses the nature of the perturbations entering our system; the *pdf*

$$p(\upsilon_k) \tag{3.10}$$

characterizes the *measurement noise*; while

$$p(\mathbf{x}_0) \tag{3.11}$$

describes our previous knowledge about the initial state and will be referred to as the *initial prior*.

[7] An uncertain control action includes both known control actions and random perturbations. Notice that this perturbation must be independent of its own past and of $\mathbf{x}_{k-1}$ in order for the process $\mathbf{x}_k$ to be Markovian.

The filtering problem can now be formulated as follows:

Problem 3.1 (The filtering problem):
Find at each time k the 'best' *estimate of the state of the system Σ conditioned to the whole historic of measurements (from initial time up to current time k).* ◇

Elements for the general solution

The solution needs the computation of the *posterior density*[8] of the system's state $\mathbf{x}_k$ conditioned to the set of measurements $\mathbf{y}_0^k \triangleq \{\mathbf{y}_0, \cdots, \mathbf{y}_k\}$:

$$p(\mathbf{x}_k|\mathbf{y}_0^k) \tag{3.12}$$

On this posterior we chose the 'best' or 'optimal' estimate $\hat{\mathbf{x}}_k$ by extremizing an arbitrary quality criterion. We briefly enumerate the two most often used criterions.

- The *a-posteriori maximum of likelihood* estimator takes the estimate that maximizes the posterior density:
$$\hat{\mathbf{x}}_k^{ML} \triangleq \underset{\mathbf{x}_k}{\text{argmax}} \big(p(\mathbf{x}_k|\mathbf{y}_0^k)\big). \tag{3.13}$$

- The *minimum of variance* estimator minimizes the estimation error in the L^2 norm sense:
$$\hat{\mathbf{x}}_k^{mV} \triangleq \underset{\mathbf{x}_k^*}{\text{argmin}}\ \mathbb{E}\big[\|\mathbf{x}_k - \mathbf{x}_k^*\|^2\big]. \tag{3.14}$$
One can show that this estimator is equivalent to the conditioned mean
$$\hat{\mathbf{x}}_k^{mV} = \mathbb{E}\big[\mathbf{x}_k|\mathbf{y}_0^k\big]. \tag{3.15}$$

The choice of the estimator to use depends on the application and may lead to different filtering techniques. However, we notice that the computation of the posterior (3.12) is itself not dependent on the chosen estimator.

To illustrate the differences between both estimators we give in Table 3.1 the estimates of the *pdfs* of Fig. 3.1. Observe how maximum of likelihood estimators may not give satisfactory results for *pdfs* not exhibiting a dominant maximum; how Gaussian densities lead exactly to the same estimates regardless of the estimator used; and how in multi-modal densities the minimum of variance estimator will often lead to absurd estimates.[9]

3.1.3 Incremental filtering

If we are to calculate the posterior (3.12) and a best estimate (3.13) or (3.14) at each time k, we have two possibilities:

1. Analyzing at each k the whole information we have from time 0 to time k; or
2. Refining the last posterior at time $k-1$ with the new information at time k.

[8] Posterior, *i.e.* after the observation.

[9] It was either this or next Saturday, but definitely not on Sunday!

Table 3.1: Comparison of maximum of likelihood and minimum of variance estimators applied to the densities of Fig. 3.1.

pdf	$\hat{\mathbf{x}}_k^{ML}$	$\hat{\mathbf{x}}_k^{mV}$
Uniform $p(x)$	Anything in $[2, 6]$	4
Gaussian $p(x)$	4	4
Gaussian $p(t)$	10:05	10:05
Bimodal $p(t)$	Anytime during first Saturday	Sunday at 21:36

It is obvious that *1)* requires processing an increasing amount of information as k increases. This may be suitable for systems that are not required on-line, where filtering may be based on data mining and post-processing. On the contrary, a filter that is able to perform *2)* has to process at each k a bounded amount of information and therefore it will be suitable for on-line, real-time operation. This second *incremental* operation, which leads to *recursive* formulations, is what we examine now.

The posterior at time k can be related to the posterior at time $k-1$ by a sequence of two differentiated mechanisms: *a)* a *prediction step* is used to propagate the posterior density at time $k-1$ into a *prior density*[10] at time k by using the knowledge on the system dynamics and the perturbations; and *b)* a *correction step* permits us to refine the predicted prior onto a corrected posterior by using the evidence provided by the measurements.

The sequence is initially fed with $p(\mathbf{x}_0)$, the initial prior (3.11). If a measurement $\mathbf{y}_0$ is available we will start by correcting this prior; otherwise we will start by predicting the next prior at $k = 1$.[11]

The prediction step

The *prediction step* (or '*time update*') gives the prior density at time k from the posterior at time $k-1$ by applying the knowledge about the system perturbed dynamics represented by equation (3.7). From the Markovian character of $\mathbf{x}_k$ this prior is obtained with the following recursive integral equation (the *Chapman-Kolmogorov* equation):

$$p(\mathbf{x}_k|\mathbf{y}_0^{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1}|\mathbf{y}_0^{k-1})\, d\mathbf{x}_{k-1} \tag{3.16}$$

where $p(\mathbf{x}_{k-1}|\mathbf{y}_0^{k-1})$ is the posterior (3.12) at time $k-1$ and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the *transition density* from time $k-1$ to time k obtained with (3.7) and (3.9).

Remark 3.1 (The prediction step is a convolution-like procedure). The nature of this operation is a convolution of *pdfs*. That is, the previous knowledge on the system is composed with the knowledge about its probable evolutions to predict the knowledge on the system one time step ahead. When these knowledge is expressed via *pdfs* the composition is made with a convolution-like procedure. A simple example is illustrated in Fig. 3.3 which shows the smoothing effect of prediction. □

[10] Prior, *i.e.* before the observation.

[11] In the absence of this measurement the initial prior becomes the posterior at $k = 0$.

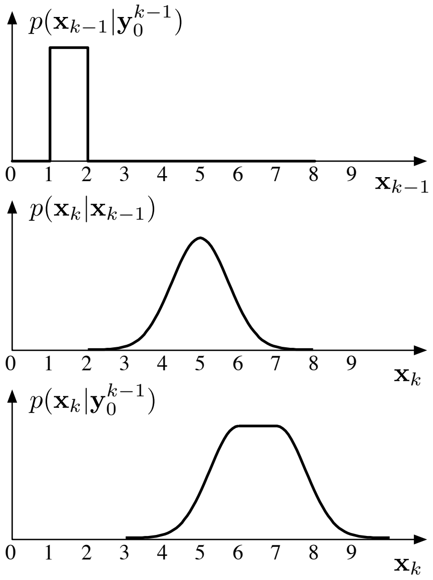

Figure 3.3: A convolution of *pdfs* performs the prediction step. The posterior at time $k-1$ says 'An object is on the second segment, between 1m and 2m'. The uncertain knowledge on the dynamics states 'It has moved about 5m to the right'. The result is the prior on bottom, which is *less accurate* than both the posterior and the perturbation.

Remark 3.2 (Analytical intractability of the prediction step). A functional calculation of the Chapman-Kolmogorov equation is, in the general case, not feasible in finite dimension. The only notable exception is the linear-Gaussian case which leads to the Kalman filter. The whole amalgam of filtering techniques are just ingenious attempts to surpass this fundamental impediment via suitable approximations. □

The correction step

The *correction step* (or '*measurement update*') gives the posterior density at time k by adding the noisy measurement information of equation (3.8). This posterior is computed with the following equation which is basically performing the *Bayes* rule

$$p(\mathbf{x}_k|\mathbf{y}_0^k) = \frac{p(\mathbf{x}_k|\mathbf{y}_0^{k-1})\,p(\mathbf{y}_k|\mathbf{x}_k)}{\int p(\mathbf{x}_k|\mathbf{y}_0^{k-1})\,p(\mathbf{y}_k|\mathbf{x}_k)\,d\mathbf{x}_k} \tag{3.17}$$

where $p(\mathbf{x}_k|\mathbf{y}_0^{k-1})$ is the prior (3.16) obtained in the prediction step and $p(\mathbf{y}_k|\mathbf{x}_k)$ is the measurement density at time k obtained with (3.8) and (3.10). The integral appearing in the denominator is not dependent on the system state and is therefore considered as a normalization factor.

Remark 3.3 (The correction step is a product). The nature of this operation is a product of *pdfs*. That is, the predicted knowledge on the system is composed with the perceived evidence to obtain a new, refined knowledge. When these knowledge is expressed via *pdfs* this composition is made with a product. The correction of the prior predicted in Fig. 3.3 is illustrated in Fig. 3.4. □

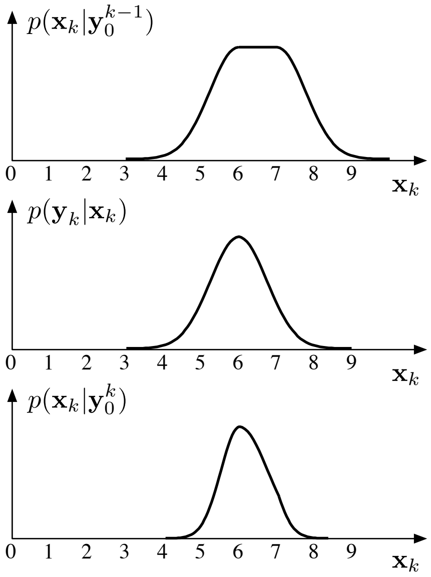

Figure 3.4: A product of *pdfs* performs the correction step. The prior of Fig. 3.3 at time k says 'The object must be between 4m and 9m'. The perceived evidence states 'I have seen it around 6m'. The result is the posterior on bottom, which is *more accurate* than both the prior and the observation.

Remark 3.4 (Additive observation noise). In the case of additive observation noise $\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k) + \upsilon_k$, which is normally a quite realistic situation, the term $p(\mathbf{y}_k|\mathbf{x}_k)$ corresponds to writing

$$p(\mathbf{y}_k|\mathbf{x}_k) = p(\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k)) = p(\upsilon_k),$$

i.e. that the *pdf* of the variable $\mathbf{y}_k - \mathbf{h}_k(\mathbf{x}_k)$ follows that of υ_k. □

3.2 The Kalman and Extended Kalman Filters

We study here the recursive solution to the filtering problem when all uncertainties are considered Gaussian. For the Kalman Filter (KF) [Kalman 1960], the additional constraint of linear evolution and observation equations leads to a finite dimension functional formulation of the whole prediciton-correction loop which is closed and provably optimal. This linearity assumption is then relaxed to construct the sub-optimal Extended Kalman Filter (EKF) [Breakwell 1967; Jazwinski 1970] via local linearizations around the most recent computed estimates.

3.2.1 The Kalman Filter

Consider the linear Gaussian system

$$\begin{aligned}
\mathbf{x}_k &= \mathbf{F}\,\mathbf{x}_{k-1} + \mathbf{G}\,\omega_k && (3.18)\\
\mathbf{y}_k &= \mathbf{H}\,\mathbf{x}_k + \upsilon_k && (3.19)\\
p(\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_0 - \bar{\mathbf{x}}_0; \mathbf{P}_0) && (3.20)\\
p(\omega_k) &= \mathcal{N}(\omega_k - 0; \mathbf{Q}) && (3.21)\\
p(\upsilon_k) &= \mathcal{N}(\upsilon_k - 0; \mathbf{R}) && (3.22)
\end{aligned}$$

where

$$\mathcal{N}(\mathbf{x}-\bar{\mathbf{x}};\mathbf{P}) \triangleq \frac{1}{\sqrt{(2\pi)^n|\mathbf{P}|}}\exp\big(-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^\top\mathbf{P}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\big) \tag{3.23}$$

is the *pdf* of an n-dimensional Gaussian variable $\mathbf{x}$ with mean $\bar{\mathbf{x}}$ and covariances matrix $\mathbf{P}$, and where $\mathbf{x}_{k-1}$, ω_k and υ_k are mutually independent. Consider also the following facts on Gaussian densities collected in the next proposition.

Proposition 3.1 (Properties of Gaussian densities). *The Gaussian density satisfies the following properties:*

1. *Any Gaussian density is uniquely identified by specifying its mean and its covariances matrix.*

2. *The density of any linear combination of Gaussian variables is Gaussian.*

3. *The product of two Gaussian densities is a non-normalized Gaussian density.*

4. *The convolution of two Gaussian densities is a non-normalized Gaussian density.*[12] □

Therefore, from the system specifications, the Gaussian properties and Remarks 3.1 and 3.3, it is quite obvious to conclude that

- The priors (3.16) and posteriors (3.17) are Gaussian for all k.

- To describe them we just need to specify their means and covariances matrices.

The KF is then the set of equations to predict and correct the mean and covariances matrix of the system's state *pdf*. The development of these equations is not given but the main directions are indicated.

KF: Prediction step

From the definitions of the mean $\bar{\mathbf{x}}$ and the covariances matrix $\mathbf{P}$ of a multidimensional variable $\mathbf{x}$

$$\begin{aligned}\bar{\mathbf{x}} &\triangleq \mathbb{E}[\mathbf{x}]\\ \mathbf{P} &\triangleq \mathbb{E}[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^\top],\end{aligned}$$

and from the linear character of the expectation operator $\mathbb{E}[\cdot]$ and the null cross-variance of independent variables $\mathbb{E}[(\mathbf{x}-\bar{\mathbf{x}})(\omega-\bar{\omega})^\top] \equiv 0$ we get quite straightfordwardly

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{F}\,\hat{\mathbf{x}}_{k-1|k-1}\\ \mathbf{P}_{k|k-1} &= \mathbf{F}\,\mathbf{P}_{k-1|k-1}\,\mathbf{F}^\top + \mathbf{G}\,\mathbf{Q}\,\mathbf{G}^\top\end{aligned}$$

where

$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &\triangleq \mathbb{E}[\mathbf{x}_k|\mathbf{y}_0^{k-1}]\\ \mathbf{P}_{k|k-1} &\triangleq \mathbb{E}[(\mathbf{x}_k-\hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k-\hat{\mathbf{x}}_{k|k-1})^\top],\end{aligned}$$

which we will write in this lighter form for clarity (the $(\cdot)^+$ notation)

$$\hat{\mathbf{x}}^+ = \mathbf{F}\,\hat{\mathbf{x}} \tag{3.24}$$

$$\mathbf{P}^+ = \mathbf{F}\,\mathbf{P}\,\mathbf{F}^\top + \mathbf{G}\,\mathbf{Q}\,\mathbf{G}^\top \tag{3.25}$$

[12] We may contrast this notable fact against Remark 3.2.

KF: Correction step

The correction step is harder to develop. The product of *pdfs* (3.17) is explicitly written and equalled to a new Gaussian. Identifying terms and applying the *matrix inversion lemma* permit us to write, in the $(\cdot)^+$ notation, the KF correction equations as

$$\begin{aligned} \mathbf{Z} &= \mathbf{H}\,\mathbf{P}\,\mathbf{H}^\top + \mathbf{R} && (3.26)\\ \mathbf{K} &= \mathbf{P}\,\mathbf{H}^\top \cdot \mathbf{Z}^{-1} && (3.27)\\ \hat{\mathbf{x}}^+ &= \hat{\mathbf{x}} + \mathbf{K}\,(\mathbf{y} - \mathbf{H}\,\hat{\mathbf{x}}) && (3.28)\\ \mathbf{P}^+ &= \mathbf{P} - \mathbf{K}\,\mathbf{Z}\,\mathbf{K}^\top && (3.29) \end{aligned}$$

where $\mathbf{Z}$ is the covariances matrix of the *innovation* $\mathbf{z} \triangleq \mathbf{y} - \mathbf{H}\,\hat{\mathbf{x}}$, *i.e.* the amount by which the measure differs from the expected measurement $\mathbf{H}\,\hat{\mathbf{x}}$. The innovation is obviously a zero-mean Gaussian $\mathcal{N}(\mathbf{z}; \mathbf{Z})$. The matrix $\mathbf{K}$ is called the *Kalman gain* and optimally corrects the prior $\mathcal{N}(\mathbf{x} - \hat{\mathbf{x}}; \mathbf{P})$ proportionally to the innovation.

Remark 3.5 (Estimator invariant). Notice that for Gaussian densities both maximum of likelihood and minimum of variance estimators, as they are defined in (3.13) and (3.14), lead to the same estimate which equals the mean of the *pdf*. This is why the *hat* notation $\hat{\mathbf{x}}$, which states for *estimate*, is confounded with the *bar* one $\bar{\mathbf{x}}$ which states for *mean*. □

3.2.2 The Extended Kalman Filter

Consider the non-linear Gaussian system

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \omega_k) && (3.30)\\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k) + \upsilon_k && (3.31)\\ p(\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_0 - \bar{\mathbf{x}}_0; \mathbf{P}_0) && (3.32)\\ p(\omega_k) &= \mathcal{N}(\omega_k - \bar{\omega}_k; \mathbf{Q}) && (3.33)\\ p(\upsilon_k) &= \mathcal{N}(\upsilon_k; \mathbf{R}) && (3.34) \end{aligned}$$

where the fact $\bar{\omega}_k \neq 0$ accounts for known control actions, that is to say the means of the perturbations.

At each time step the EKF linearizes the non-linear functions around the most recent best estimate (more details on function linearization can be found in Appendix A). Then it blindly applies the KF equations on the linearized model.

EKF: Prediction step

The evolution equation is linearized around the last best estimate and the known control action with respect to the system state and the perturbation via the Jacobian matrices

$$\mathbf{F}_\mathbf{x} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top}\right|_{\hat{\mathbf{x}},\bar{\omega}} \qquad \mathbf{F}_\omega = \left.\frac{\partial \mathbf{f}}{\partial \omega^\top}\right|_{\hat{\mathbf{x}},\bar{\omega}}$$

This leads to the EKF prediction equations

$$\begin{aligned} \hat{\mathbf{x}}^+ &= \mathbf{f}(\hat{\mathbf{x}}, \bar{\omega}) && (3.35)\\ \mathbf{P}^+ &= \mathbf{F}_\mathbf{x}\,\mathbf{P}\,\mathbf{F}_\mathbf{x}^\top + \mathbf{F}_\omega\,\mathbf{Q}\,\mathbf{F}_\omega^\top && (3.36) \end{aligned}$$

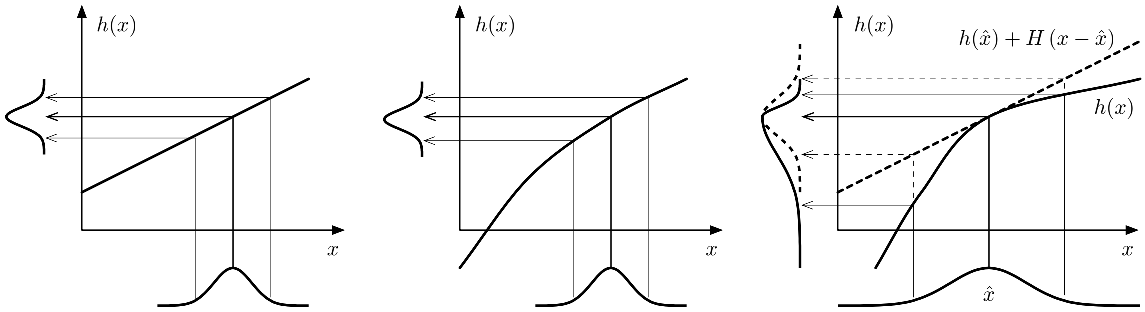

Figure 3.5: Contrasting non-linearities against Gaussian uncertainties in EKF. *Left:* Linear system. *Center:* Non-linear system is almost linear inside the region of confidence of the Gaussian. *Right:* Too large uncertainties with respect to the linearization validity margin will destroy the Gaussian approximation assumption at the output: EKF will perform the operation sketched in dashed line while the true density propagation should follow the solid one. This will lead to biased estimates that may make the filter diverge.

EKF: Correction step

The measurement equation is linearized around the last best estimate with respect to the system state via the Jacobian matrix

$$\mathbf{H} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\top}\right|_{\hat{\mathbf{x}}}.$$

This leads to the EKF correction equations

$$\mathbf{Z} = \mathbf{H}\,\mathbf{P}\,\mathbf{H}^\top + \mathbf{R} \tag{3.37}$$

$$\mathbf{K} = \mathbf{P}\,\mathbf{H}^\top \cdot \mathbf{Z}^{-1} \tag{3.38}$$

$$\hat{\mathbf{x}}^+ = \hat{\mathbf{x}} + \mathbf{K}\,(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})) \tag{3.39}$$

$$\mathbf{P}^+ = \mathbf{P} - \mathbf{K}\,\mathbf{Z}\,\mathbf{K}^\top \tag{3.40}$$

where the innovation is now defined as $\mathbf{z} \triangleq \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})$ with covariances matrix $\mathbf{Z} = \mathbf{H}\,\mathbf{P}\,\mathbf{H}^\top + \mathbf{R}$. When EKF performs properly (see next remark), the innovation is close to a zero-mean Gaussian $\mathcal{N}(\mathbf{z};\mathbf{Z})$.

Remark 3.6 (Validity of the linearizations). The key aspect for EKF to perform adequately is that the linear approximations must hold for the whole region of confidence of the Gaussian (Fig. 3.5). We may define this region as the 2σ or 3σ ellipsoid (Appendix B). This requires the Jacobian matrices to be fairly constant inside these ellipsoids. Seen the other way around, the Gaussians need to be small enough with respect to the linear approximations, and for all k, that is, they must be initially small and they must be kept small during the filter's lifetime. □

3.3 The Gaussian Sum Filter

The Gaussian Sum Filter (GSF) [Alspach and Sorenson 1972] may be viewed as a further extension of the EKF. The main idea behind it is the approximation of the *pdfs* by Gaussian mixtures, *i.e.* sums of weighted Gaussians, each one of them adequately fulfilling the EKF

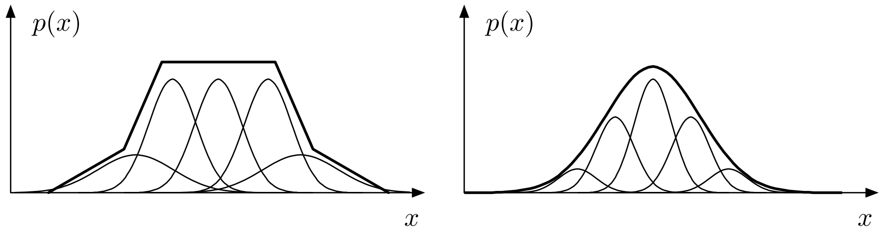

Figure 3.6: Gaussian sum approximations. *Left:* A non-Gaussian *pdf*. *Right:* A Gaussian *pdf* which is too large for the non-linearities. GSF will normally use both types of approach.

requirements on linearization. In brief, each Gaussian will be taken in charge by an EKF, for which just an additional operation will be needed to adequately up-date the Gaussian weights during predictions and corrections.

There are mainly two obvious reasons to use Gaussian sums (Fig. 3.6): *a)* the knowledge we want to represent is non-Gaussian; and *b)* the non-linearities are too severe for a single Gaussian representation. In any of these cases the problem to be solved by GSF arises from considering the following non-linear, non-Gaussian system where the non-Gaussian character of the *pdfs* has been approximated by Gaussian sums:[13]

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \omega_k) \tag{3.41}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \upsilon_k \tag{3.42}$$

$$p(\mathbf{x}_0) = \sum_{i=1}^{N_0} \rho_i \, \mathcal{N}(\mathbf{x}_0 - \bar{\mathbf{x}}_0^i; \mathbf{P}_0^i) \tag{3.43}$$

$$p(\omega_k) = \sum_{m=1}^{M} \mu_m \, \mathcal{N}(\omega_k - \bar{\omega}_k^m; \mathbf{Q}^m) \tag{3.44}$$

$$p(\upsilon_k) = \mathcal{N}(\upsilon_k - 0; \mathbf{R}) \tag{3.45}$$

GSF: Prediction step

Similarly to what we did for the Kalman filter, we consider the Gaussian properties (in Proposition 3.1) and the fact that each one of them is now respecting the linearization limits of EKF (recall Remark 3.6) to allow us to write all priors and posteriors as Gaussian sums. Assume then that at time $k-1$ we have the following posterior written with the $(\cdot)^+$ notation

$$p(\mathbf{x}|\mathbf{y}_0^{k-1}) = \sum_{i=1}^{N} \rho_i \, \mathcal{N}(\mathbf{x} - \bar{\mathbf{x}}_i; \mathbf{P}_i).$$

The prediction step convolutes this posterior with the perturbation (3.44). After that, the number of Gaussians is augmented to $N^+ = (N \cdot M)$. The obtained prior at time k is then the new sum of Gaussians

$$p(\mathbf{x}^+|\mathbf{y}_0^{k-1}) = \sum_{j=1}^{N^+} \rho_j^+ \, \mathcal{N}(\mathbf{x}^+ - \bar{\mathbf{x}}_j^+; \mathbf{P}_j^+) \tag{3.46}$$

[13] Notice that the observation noise is purely Gaussian. This is not a requirement but, as indicated before, an additive Gaussian noise is sufficiently realistic in most cases.

where its parameters are obtained as follows: for every $1 \leq i \leq N$ and $1 \leq m \leq M$ define a Gaussian $j = (M \cdot (i-1) + m)$ with

$$\begin{aligned} \rho_j^+ &= \frac{\rho_i\,\mu_m}{\sum_{i,m}(\rho_i\,\mu_m)} && (3.47)\\ \bar{\mathbf{x}}_j^+ &= \mathbf{f}(\bar{\mathbf{x}}_i, \bar{\omega}_m) && (3.48)\\ \mathbf{P}_j^+ &= \mathbf{F}_\mathbf{x}^j\,\mathbf{P}_i\,(\mathbf{F}_\mathbf{x}^j)^\top + \mathbf{F}_\omega^j\,\mathbf{Q}_m\,(\mathbf{F}_\omega^j)^\top && (3.49)\end{aligned}$$

where the Jacobian matrices are evaluated at the means of the Gaussians $\bar{\mathbf{x}}_i$ and the control actions $\bar{\omega}_m$:

$$\mathbf{F}_\mathbf{x}^j = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top}\right|_{\bar{\mathbf{x}}_i, \bar{\omega}_m} \qquad \mathbf{F}_\omega^j = \left.\frac{\partial \mathbf{f}}{\partial \omega^\top}\right|_{\bar{\mathbf{x}}_i, \bar{\omega}_m}.$$

The prediction equations simply say that each Gaussian i has given place to M Gaussians j that evolved like in an EKF according to the perturbation $\mathcal{N}\{\bar{\omega}_m; \mathbf{Q}_m\}$, getting a new weight ρ_j which is the product of its previous weight ρ_i times the weight μ_m of the perturbation.

GSF: Correction step

The correction step multiplies the prior (3.46) by the observation. In this case, the observation is a single Gaussian and therefore the number of terms of the Gaussian sum is kept unchanged. The posterior is then the sum of Gaussians

$$p(\mathbf{x}^+|\mathbf{y}_0^k) = \sum_{i=1}^{N} \rho_i^+\,\mathcal{N}(\mathbf{x}^+ - \mathbf{x}_i^+; \mathbf{P}_i^+).$$

Its terms are computed for $1 \leq i \leq N$ as follows

$$\begin{aligned} \rho_i^+ &= \frac{\lambda_i\,\rho_i}{\sum_i(\lambda_i\,\rho_i)} && (3.50)\\ \mathbf{K}_i &= \mathbf{P}_i\,\mathbf{H}_i^\top \cdot \mathbf{Z}_i^{-1} && (3.51)\\ \bar{\mathbf{x}}_i^+ &= \bar{\mathbf{x}}_i + \mathbf{K}_i(\mathbf{y} - \mathbf{h}(\bar{\mathbf{x}}_i)) && (3.52)\\ \mathbf{P}_i^+ &= \mathbf{P}_i - \mathbf{K}_i\,\mathbf{Z}_i\,\mathbf{K}_i^\top && (3.53)\end{aligned}$$

with

$$\mathbf{H}_i = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\top}\right|_{\bar{\mathbf{x}}_i}$$

where the innovations

$$\begin{aligned} \mathbf{z}_i &= \mathbf{y} - \mathbf{h}(\bar{\mathbf{x}}_i)\\ \mathbf{Z}_i &= \mathbf{H}_i\,\mathbf{P}_i\,\mathbf{H}_i^\top + \mathbf{R}\end{aligned}$$

permit us to define the terms λ_i updating the weights, which we name *likelihoods*:

$$\lambda_i \triangleq \mathcal{N}(\mathbf{z}_i; \mathbf{Z}_i) = \frac{1}{\sqrt{(2\pi)^n|\mathbf{Z}_i|}} \exp\big(-\frac{1}{2}\mathbf{z}_i{}^\top \mathbf{Z}_i^{-1}\,\mathbf{z}_i\big). \qquad (3.54)$$

We give some intuition on the notion of likelihood between a predicted prior and an observation (Fig. 3.7). The likelihood comes from considering the normalization factor in the

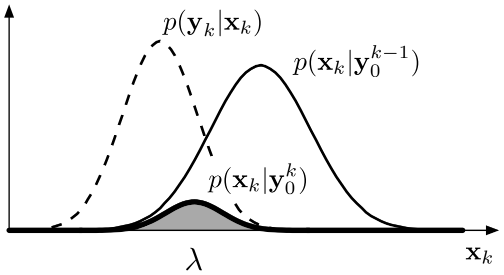

Figure 3.7: The measure of likelihood between two Gaussians. When one of them is the prior (*solid*) and the other one the observation (*dashed*), their product is the non-normalized posterior (*thick solid*). The likelihood λ is proportional to the area under this posterior.

correction equation (3.17). In effect, this normalization is done by dividing the product of densities by its integral, *i.e.* by the area under the product. This area, which is bigger when both Gaussians are close and similar, gets the name *likelihood* and can be shown to respond to expression (3.54). In multi-Gaussian representations we are interested in keeping track of all the different weighted areas $\rho_i\lambda_i$ until a global normalization is done by dividing by the sum of all of them $\sum(\rho_i\lambda_i)$ (equation (3.50)). The likelihoods evaluate how likely each Gaussian in the sum is, given the observation. When all likelihoods are normalized, λ_i may also be viewed as a measure of probability, *i.e.* the probability of Gaussian i to truly represent the system state given the observation. Therefore it makes sense to use it to modify the Gaussian weight as we do.

GSF: Estimation

After each update an estimate is calculated from the posterior to provide a comprehensive output. For the minimum of variance estimator we said the estimate is the mean of the posterior. When the weights are normalized $\sum \rho_i = 1$ this mean is easily obtained with:

$$\hat{\mathbf{x}}^{mV} = \sum_i \rho_i \, \bar{\mathbf{x}}_i.$$

The maximum of likelihood estimator takes the argument of the maximum value of the posterior. To avoid the exact computation of this maximum it is sometimes reasonable to approximate it with the mean of the Gaussian with the biggest height.[14] This is obtained with

$$\begin{aligned} i_{ML} &= \underset{i}{\operatorname{argmax}} \ \frac{\rho_i}{\sqrt{|\mathbf{P}_i|}} \\ \hat{\mathbf{x}}^{ML} &\approx \bar{\mathbf{x}}_{i_{ML}} \end{aligned}$$

GSF: Merging and pruning

As we have seen the number of Gaussians in the posterior increases geometrically with time following the recursive equation $N^+ = N \cdot M$. This leads after a short sequence of prediction-

[14] This approximation should be used with care in cases of a significant amount of measurement noise as the observations will not discriminate between nearby Gaussians.

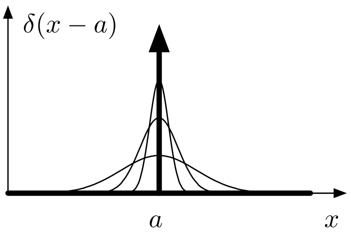

Figure 3.8: The Dirac function representing a particle compared to Gaussians of different variance.

correction steps to a filtering problem of intractable size. The number of Gaussians in the sum can be kept low by using the mechanisms of *merging* and *pruning*: *a)* Merging is the mechanism by which two similar Gaussians are combined into a single one which properly approximates the density of the formers. *b)* Pruning is the mechanism by which Gaussians with very low weight are simply deleted from the sum.

These and other possible facts on the implementation of the GSF are not given here.

3.4 The Particle Filter

The Particle Filter (PF) [Doucet et al. 2001] is a further step in the approximation of the posterior, where now a sum of Dirac functions or *particles* is used. With respect to this approximation, we will consider the PF as a special case of the GSF in the sense that a Dirac function is no less than a Gaussian with zero variance (Fig. 3.8):

$$\delta(x-a) \triangleq \lim_{\sigma\to 0} \mathcal{N}(x-a;\sigma^2)$$

Proposition 3.2 (Properties of the Dirac function). *A Dirac function has the following properties:*

1. *Normalized weight*
$$\int \delta(x-a)\,dx = 1$$

2. *Multiplication by a function which is defined in the Dirac base point*
$$f(x)\,\delta(x-a) = f(a)\,\delta(x-a)$$
This gives as a consequence
$$\int f(x)\,\delta(x-a)\,dx = f(a).$$

3. *Pure function translation after convolution*
$$f(x) * \delta(x-a) = f(x-a)$$

4. *A function $y = f(x)$ of a Dirac-distributed variable $p(x) = \delta(x-a)$ is a Dirac-distributed variable with* pdf
$$p(y) = \delta(y - f(a)).$$
□

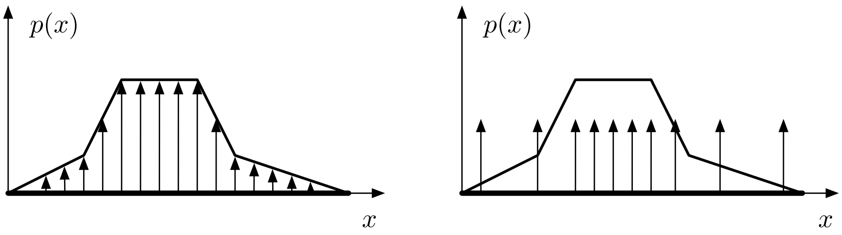

Figure 3.9: Particle approximations of a *pdf*. *Left:* Uniformly spaced, unevenly weighted particles. *Right:* Uniformly weighted, unevenly spaced particles. The PF keeps the density approximation up-to-date by using both uneven weighting and spacing.

The punctual character of the particles is naturally unaffected by non-linearities. This will provide notable advantages in flexibility and simplicity of formulation as we will see. The price we have to pay is the necessity of using a very large number of –punctual– particles to properly approximate the –naturally continuous– densities (Fig. 3.9).

Consider the non-linear, non-Gaussian system with the initial prior approximated by a weighted sum of particles and a whatever perturbation density:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \omega_k) \tag{3.55}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \upsilon_k \tag{3.56}$$

$$p(\mathbf{x}_0) = \sum_{i=1}^{N} \rho_i \, \delta(\mathbf{x}_0 - \mathbf{x}_0^i) \tag{3.57}$$

$$p(\omega_k) = \mathit{whatever} \tag{3.58}$$

$$p(\upsilon_k) = \mathcal{N}(\upsilon_k - 0; \mathbf{R}) \tag{3.59}$$

PF: Prediction step

The radical novelty of the PF resides in the prediction stage, where each particle is considered a deterministic realization of the system[15] and is evolved exactly as the evolution equation states. For this, a different perturbation, randomly generated following precisely the perturbation distribution, is deterministically imposed to each of the particles. This avoids computing the prediction convolution and guarantees that the obtained prior is also a sum of particles.

We consider the posterior at time $k-1$ to be a weighted sum of particles

$$p(\mathbf{x}|\mathbf{y}_0^{k-1}) = \sum_{i=1}^{N} \rho_i \, \delta(\mathbf{x} - \mathbf{x}_i).$$

We have to generate a set of N independent perturbations that mimic the perturbation distribution

$$\{\omega_1, \cdots, \omega_N\} \quad \text{so that} \quad p(\{\omega_1, \cdots, \omega_N\}) \sim p(\omega).$$

[15] Notice that $p_X(x) = \delta(x-a)$ means $X = a$ with probability 1, that is X is deterministic and exactly equal to a.

Then the particles are evolved deterministically with their corresponding perturbations. Their weights are unchanged:

$$\rho_i^+ = \rho_i \tag{3.60}$$

$$\mathbf{x}_i^+ = \mathbf{f}(\mathbf{x}_i, \omega_i) \tag{3.61}$$

PF: Correction step

The correction stage follows a similar scheme to that of the GSF, where the observation noise is also considered additive and Gaussian.[16] As the product of a sum of particles with any other function gives a new sum of particles, the obtained posterior continues to be a sum of particles.

The particles are just re-weighted according to its likelihood with the observation. Their positions are unchanged:

$$\rho_i^+ = \frac{\lambda_i\,\rho_i}{\sum_i \lambda_i\,\rho_i} \tag{3.62}$$

$$\mathbf{x}_i^+ = \mathbf{x}_i \tag{3.63}$$

where the likelihoods λ_i are now[17]

$$\lambda_i = \mathcal{N}(\mathbf{y} - \mathbf{h}(\mathbf{x}_i); \mathbf{R}). \tag{3.64}$$

PF: Estimation

The estimates in the PF are also very straightforwardly computed. The minimum of variance estimator is

$$\hat{\mathbf{x}}^{mV} = \sum_i \rho_i\,\mathbf{x}_i.$$

And the maximum of likelihood is

$$\begin{aligned} i_{ML} &= \operatorname*{argmax}_i \rho_i \\ \hat{\mathbf{x}}^{ML} &= \mathbf{x}_{i_{ML}}. \end{aligned}$$

PF: Particle redistribution

As we have seen the particles only move in the prediction stage; they do it randomly as described by the perturbation density, and this behavior is not compensated in the correction step, which only performs weight updates. This generates an increasing dispersion of the particles, to the point that after some time they are so sparsely distributed that they can no longer approximate the state density (Fig. 3.10). When weighted by the observations, those

[16] This is again not a requirement. See note 13.

[17] The likelihood definition given in (3.54) for the GSF does not change; it is just that for particles we have $\mathbf{P}_i \equiv 0$ and hence the innovation covariances matrix becomes $\mathbf{Z}_i \triangleq \mathbf{H}_i\mathbf{P}_i\mathbf{H}_i^\top + \mathbf{R} = \mathbf{R}$.

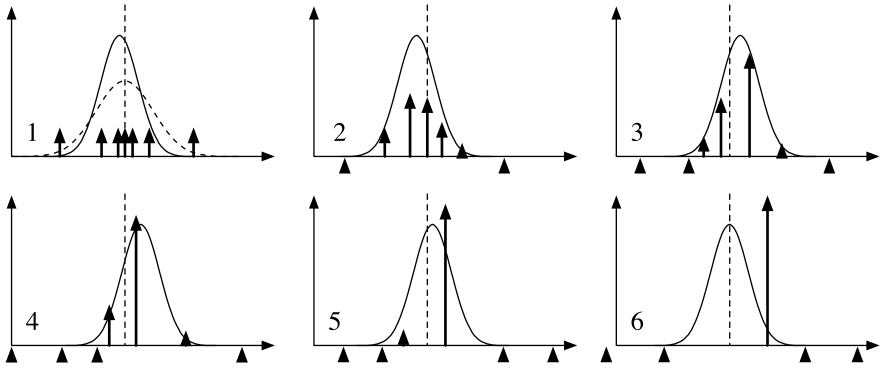

Figure 3.10: Particle degeneracy: a progressive degradation of the particle sum approximation. The true state is at the vertical dotted line, with an initial prior indicated by the dotted Gaussian in 1, which is approximated by the particle sum as shown. The noisy observations are represented by the solid Gaussian.

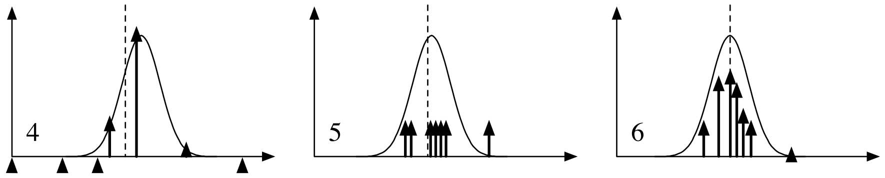

Figure 3.11: Particle redistribution must be applied when the distribution starts to degenerate, for example at time 4 in the sequence of Fig. 3.10. Compare the new distributions at times 5 and 6 with those in that figure.

particles that lie far apart from the true density will get weights closer and closer to zero. One particle will accumulate the whole weight, but it will also diverge from the true state due to the random dynamics imposed. We speak about *particle degeneracy*, a phenomenon that must be avoided.

Particle redistribution (Fig. 3.11) corrects this effect and renders the PF provably convergent. Weak particles are eliminated and strong ones are reproduced. Briefly, for every particle $\{\rho_i; \mathbf{x}_i\}$ we generate a number of $n = \text{round}(\rho_i \cdot N)$ particles which share the same position $\mathbf{x}_i$ and get a new weight $1/N$. This way *a)* the probability density is unchanged and *b)* the particles are regrouped where the system state is more likely to be.

Alternative solutions to the degeneracy problem are also possible. These and other facts on the implementation of the PF are not given here. The interested reader is suggested to consult for example [Doucet et al. 2001].

Chapter 4

Simultaneous Localization And Mapping

4.1 Introduction

This chapter takes the material from the precedent chapters to build a system that, based on vision or on other exteroceptive sensors, allows a robot to construct a map of *landmarks* of its surrounding world, which are used at the same time to get localized in it. The set of procedures to set up such a system are known by the acronym SLAM, that accounts for 'Simultaneous Localization And Mapping'.

In the large sense, SLAM is as old as humanity, or even older: any task of exploration that tries to memorize the particularities of the explored area, so that the explorer can have an idea of it as a whole as-well as of where he is, is inherently taking a SLAM approach. With this in mind, it seems reasonable to highlight the following requirements for such a system:

1. **Static world.** What we map is what we are going to use, at the same or at a later time, to get localized. Objects or landmarks that move or that are not stable in some way should be ignored.

2. **Incremental operation.** The mapping and the self-localization are performed at the same time of the exploration: the map and the explorer's localization must be always up-to-date, because failing to do so would automatically result in the explorer getting lost. Every new information is incrementally used either to add some new landmark in the map or to correct both the map and the localization.

Depending on the performance of the tools we use we will be able to explore larger areas and build more accurate maps.

In a more restricted sense, and in spite of what we said above about the SLAM foundations, the term 'SLAM' itself belongs to the robotics community, and therefore it is just some decades old. It is when the explorer is a machine, a robot for example, when the whole material of the previous chapters enters the game. SLAM must put into play robust and scalable real-time algorithms that fall into three main categories: *estimation* (in the prediction-correction sense), *perception* (with its signal processing), and *decision* (what we could call *strategy*).

The **estimation** side is normally solved by filtering techniques as illustrated in Fig. 4.1. The problem is now well understood: the whole decade of the nineties was devoted to solve it

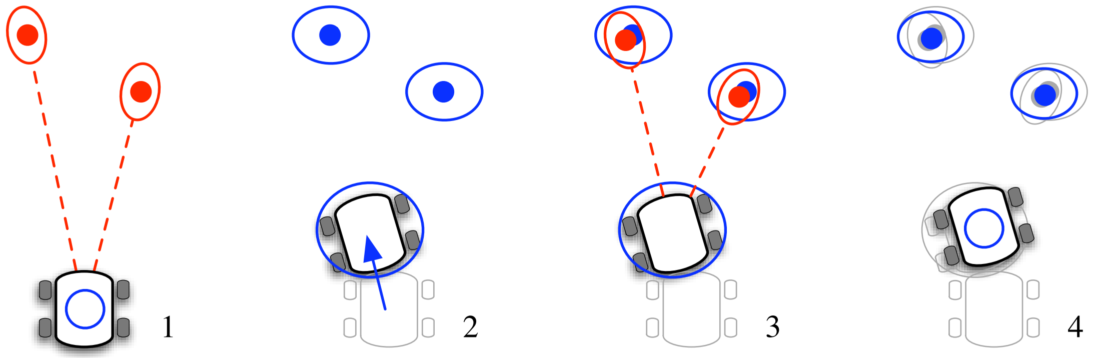

Figure 4.1: Typical SLAM operations during a complete filter loop. *1*: perception of new landmarks and their initialization in the map. *2*: Prediction of robot motion with associated increase of its position uncertainty. *3*: Observation of already mapped landmarks from an uncertain robot position. *4*: Correction of landmark positions and robot localization, with associated decrease of both robot and map uncertainties. Ellipses represent the uncertainty boundaries.

in 2D with range and bearing sensors, and big progress was achieved in this estimation side —to the point that some claim that the subject is approaching saturation, which is a kind of optimality obtained by evolutionary mechanisms— and recent research has focused on the perception side.

The **perception** side (or the problem of landmark detection, recognition and measuring) is considered, in this chapter, to be solved. Landmark measurements are considered to be in the *range-and-bearing* format, *i.e.* the robot measures the directions and the distances to the landmarks with respect to its own position. The problem of perception, which is now one of the bottlenecks in SLAM, will be more accurately studied in the second part of this thesis for the case of vision.

The **decision** side should cover questions of the kind “where do I move next”, “how many and which landmarks do I observe”, “when and where do I initialize a new landmark”, etc. Its relevance is often minimized (even ignored!) and solved by a set of heuristic strategies, although some approaches exist to formalize strategies with the aim to maximize the information gain or the explored area.

This chapter is dedicated mainly to revise the original and one of the most popular SLAM estimation solutions: Extended Kalman Filter SLAM. More performing alternatives are also briefly introduced: the Extended Information Filter SLAM, and FastSLAM2.0 which is based on the Particle Filter.

4.2 Extended Kalman Filter SLAM

The first consistent SLAM algorithm dates back from 1987 and is due to Smith and Cheeseman [1987]. They considered a 2D world (although we give a dimensionally independent formulation) and a robot equipped with a range and bearing sensor (a laser range scanner) equipped with some ego-motion sensing based on odometry. The fusion engine was a simple Extended Kalman Filter.

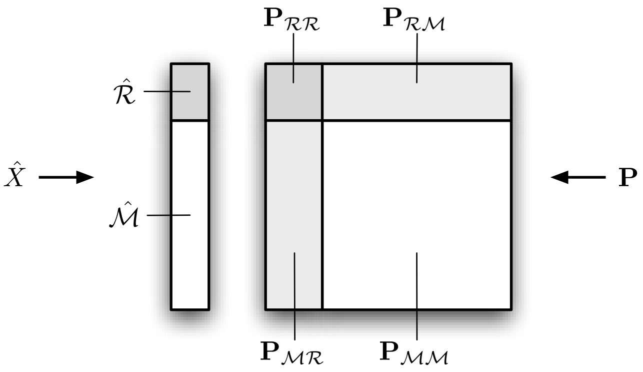
$\mathbf{P}_{\mathcal{RR}}$
$\mathbf{P}_{\mathcal{RM}}$
$\hat{\mathcal{R}}$
$\hat{X}$
$\hat{\mathcal{M}}$
$\mathbf{P}$
$\mathbf{P}_{\mathcal{MR}}$
$\mathbf{P}_{\mathcal{MM}}$

Figure 4.2: The EKF-SLAM map with robot pose and landmarks means, covariances and cross-variances.

4.2.1 The algorithm

In EKF-SLAM, the *map* consists of a random state vector containing the robot pose and the currently mapped landmark positions:

$$X = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} \tag{4.1}$$

with

$$\mathcal{R} = \begin{bmatrix} \mathbf{x} \\ \mathbf{q} \end{bmatrix} \qquad \text{and} \qquad \mathcal{M} = \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_n \end{bmatrix}$$

where $\mathcal{R}$ is the robot state containing position and orientation and $\mathcal{M}$ is the set of landmark positions, all expressed in the same, global reference frame. In the EKF framework, the *a posteriori* density is approximated by a Gaussian density with mean and covariances matrix defined by

$$\hat{X} = \begin{bmatrix} \hat{\mathcal{R}} \\ \hat{\mathcal{M}} \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{P}_{\mathcal{RR}} & \mathbf{P}_{\mathcal{RM}} \\ \mathbf{P}_{\mathcal{MR}} & \mathbf{P}_{\mathcal{MM}} \end{bmatrix}, \tag{4.2}$$

where the fact of the covariances matrix $\mathbf{P}$ being symmetric implies $\mathbf{P}_{\mathcal{RM}} = \mathbf{P}_{\mathcal{MR}}^\top$. Such a map representation is illustrated in Fig. 4.2.

The objective of the SLAM system is to keep this *pdf* up-to-date when any of the following situations occurs:

1. The robot moves;
2. The robot perceives a landmark already existing in the map; and
3. The robot perceives a new landmark and decides to incorporate in the map.

These operations are described in the following paragraphs.

EKF-SLAM: Robot motion: the prediction step

The evolution of the robot pose during one time step is described by the function

$$\mathcal{R}^+ = \mathbf{f}(\mathcal{R}, \mathbf{u}) \tag{4.3}$$

where $\mathbf{u} \sim \mathcal{N}\{\hat{\mathbf{u}}; \mathbf{U}\}$ is a vector of controls assumed to be Gaussian with mean $\hat{\mathbf{u}}$ and covariances matrix $\mathbf{U}$. From the EKF formulation we get the prediction step

$$\begin{aligned} \hat{\mathcal{R}}^+ &= \mathbf{f}(\hat{\mathcal{R}}, \hat{\mathbf{u}}) && (4.4) \\ \mathbf{P}^+_{\mathcal{R}\mathcal{R}} &= \mathbf{F}_\mathcal{R} \mathbf{P}_{\mathcal{R}\mathcal{R}} \mathbf{F}_\mathcal{R}^\top + \mathbf{F}_\mathbf{u} \mathbf{U} \mathbf{F}_\mathbf{u}^\top && (4.5) \\ \mathbf{P}^+_{\mathcal{R}\mathcal{M}} &= \mathbf{F}_\mathcal{R} \mathbf{P}_{\mathcal{R}\mathcal{M}} && (4.6) \\ \mathbf{P}^+_{\mathcal{M}\mathcal{M}} &= \mathbf{P}_{\mathcal{M}\mathcal{M}} && (4.7) \end{aligned}$$

where the Jacobian matrices are defined by

$$\mathbf{F}_\mathcal{R} = \left.\frac{\partial \mathbf{f}}{\partial \mathcal{R}^\top}\right|_{\hat{\mathcal{R}}, \hat{\mathbf{u}}} \qquad \mathbf{F}_\mathbf{u} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}^\top}\right|_{\hat{\mathcal{R}}, \hat{\mathbf{u}}} .$$

EKF-SLAM: Observations of existing landmarks: the correction step

The measure of a landmark i is described by the function

$$\mathbf{y}_i = \mathbf{h}(\mathcal{R}, \mathbf{p}_i) + \upsilon \qquad (4.8)$$

where $\upsilon \sim \mathcal{N}\{0; \mathbf{R}\}$ is a white Gaussian noise with covariances matrix $\mathbf{R}$. From the EKF formulation we get the correction step at observation of landmark i

$$\begin{aligned} \mathbf{z}_i &= \mathbf{y}_i - \mathbf{h}(\hat{\mathcal{R}}, \hat{\mathbf{p}}_i) && (4.9) \\ \mathbf{Z}_i &= \mathbf{H}_i \, \mathbf{P} \, \mathbf{H}_i^\top + \mathbf{R} && (4.10) \\ \mathbf{K}_i &= \mathbf{P} \, \mathbf{H}_i^\top \cdot \mathbf{Z}_i^{-1} && (4.11) \\ \hat{X}^+ &= \hat{X} + \mathbf{K}_i \cdot \mathbf{z}_i && (4.12) \\ \mathbf{P}^+ &= \mathbf{P} - \mathbf{K}_i \, \mathbf{Z}_i \, \mathbf{K}_i^\top && (4.13) \end{aligned}$$

where the Jacobian matrix is defined by

$$\mathbf{H}_i = \left.\frac{\partial \mathbf{h}(\mathcal{R}, \mathbf{p}_i)}{\partial X^\top}\right|_{\hat{X}}$$

and where it may be worth noticing that $\frac{\partial \mathbf{h}}{\partial \mathbf{p}_j{}^\top} = \mathbf{0}$ for $j \neq i$ and thus that $\mathbf{H}_i$ is sparse:

$$\mathbf{H}_i = \begin{bmatrix} \left.\frac{\partial \mathbf{h}}{\partial \mathcal{R}^\top}\right|_{\hat{\mathcal{R}}, \hat{\mathbf{p}}_i} & \mathbf{0} & \cdots & \mathbf{0} & \left.\frac{\partial \mathbf{h}}{\partial \mathbf{p}_i{}^\top}\right|_{\hat{\mathcal{R}}, \hat{\mathbf{p}}_i} & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} .$$

EKF-SLAM: Landmark initialization

Initialization consists of stacking the new landmark position $\mathbf{p}$ into the map as

$$X^+ = \begin{bmatrix} X \\ \mathbf{p} \end{bmatrix} \qquad (4.14)$$

and defining the *pdf* of this new state (the resulting map) conditioned to observation $\mathbf{y}$. This task is easily performed from the first observation given by $\mathbf{y} = \mathbf{h}(\mathcal{R}, \mathbf{p}) + \upsilon$ as all the components of $\mathbf{p}$ are observed. The classic method [Newman 1999] performs the variable change

$$\mathbf{w} = \mathbf{h}(\mathcal{R}, \mathbf{p}) \qquad (4.15)$$

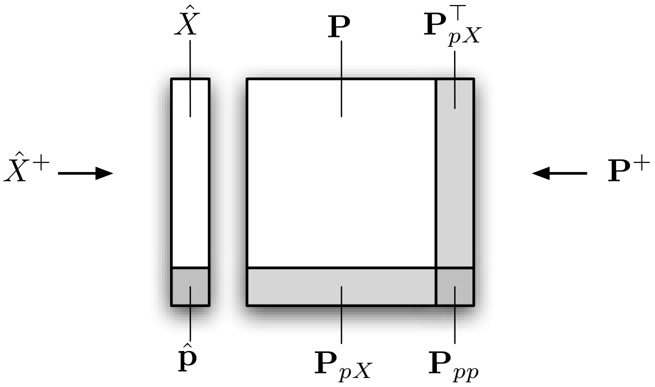

Figure 4.3: The EKF-SLAM map after landmark initialization.

so the measurement is now $\mathbf{y} = \mathbf{w} + \upsilon$. Then it defines the function $\mathbf{g}$, inverse of $\mathbf{h}$, in order to obtain an explicit expression of $\mathbf{p}$

$$\mathbf{p} = \mathbf{g}(\mathcal{R}, \mathbf{w}). \tag{4.16}$$

Assuming that $\mathbf{P}_{\mathcal{R}\mathcal{R}}$ and $\mathbf{R}$ are small enough we can approximate this expression with the Taylor series truncated at the linear terms

$$\mathbf{p} \approx \mathbf{g}(\hat{\mathcal{R}}, \mathbf{y}) + \mathbf{G}_{\mathcal{R}}(\mathcal{R} - \hat{\mathcal{R}}) + \mathbf{G}_{\mathbf{w}}(\mathbf{w} - \mathbf{y}) \tag{4.17}$$

with the Jacobian matrices defined by

$$\mathbf{G}_{\mathcal{R}} = \left.\frac{\partial \mathbf{g}}{\partial \mathcal{R}^\top}\right|_{\hat{\mathcal{R}},\mathbf{y}} \qquad \mathbf{G}_{\mathbf{w}} = \left.\frac{\partial \mathbf{g}}{\partial \mathbf{w}^\top}\right|_{\hat{\mathcal{R}},\mathbf{y}}$$

Then $\mathbf{p}$ can be considered approximately Gaussian with mean and covariances matrices defined by

$$\hat{\mathbf{p}} = \mathbf{g}(\hat{\mathcal{R}}, \mathbf{y}) \tag{4.18}$$

$$\mathbf{P}_{pX} = \mathbf{G}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}X} \tag{4.19}$$

$$\mathbf{P}_{pp} = \mathbf{G}_{\mathcal{R}}\mathbf{P}_{\mathcal{R}\mathcal{R}}\mathbf{G}_{\mathcal{R}}^\top + \mathbf{G}_{\mathbf{w}}\mathbf{R}\mathbf{G}_{\mathbf{w}}^\top \tag{4.20}$$

where $\mathbf{P}_{\mathcal{R}X} = \begin{bmatrix}\mathbf{P}_{\mathcal{R}\mathcal{R}} & \mathbf{P}_{\mathcal{R}\mathcal{M}}\end{bmatrix}$ (see Fig. 4.2). The augmented map is finally specified by

$$\hat{X}^+ = \begin{bmatrix}\hat{X} \\ \hat{\mathbf{p}}\end{bmatrix} \qquad \mathbf{P}^+ = \begin{bmatrix}\mathbf{P} & \mathbf{P}_{pX}^\top \\ \mathbf{P}_{pX} & \mathbf{P}_{pp}\end{bmatrix}. \tag{4.21}$$

which can be appreciated in Fig. 4.3.

4.2.2 Algorithm complexity

We evaluate the algorithm complexity as a function of the number of landmarks n. The prediction step has linear complexity $O(n)$ because of (4.6); the correction step has quadratic complexity $O(n^2)$ because of (4.13); and the initialization step is $O(n)$ because of (4.19). When a number k of landmarks is observed at each time step, the overall complexity of EKF-SLAM is $O(k\,n^2)$. This limits the usage of EKF-SLAM to moderately small maps. To map larger or denser areas the solution of building smaller sub-maps that are linked together to keep consistency is proposed in [Estrada et al. 2005]. By restricting every sub-map to a fixed maximum size, this solution can be considered $O(1)$, *i.e.* of constant-time complexity.

4.3 Scalable SLAM algorithms

The poor performances of EKF-SLAM in terms of algorithmic complexity together with the inherent linearization issues of EKF have triggered an impressive amount of research on the filtering side of SLAM. The aim of every real-time algorithm should be to achieve a constant time complexity so that it can run for as long time as required without showing signs of saturation. Besides the aforementioned work by Estrada et al., we briefly enunciate the two solutions that achieved this objective.

4.3.1 Exactly-Sparse Extended Information Filter SLAM

A remarkable step towards constant-time SLAM is due to Thrun et al. [2004]. It makes use of the natural quasi-sparsity of the information matrix in the Extended Information Filter[1] when applied to SLAM problems. This sparsity is only polluted by the cross-correlations between the robot pose and the set of landmarks that are generated when applying a filter time-step with uncertain robot motion. This pollution affects notably those landmarks that lie close to the robot, *i.e.* close to those other landmarks that have been recently observed, and it practically leaves the rest of the information matrix entries untouched. Sparse Extended Information Filter SLAM approximates these entries with the null value and then considers only a constant-size sub-set of the problem to perform the updates. This permits to complete the filter loop in constant time.

Further improvements to this algorithm, *e.g.* [Eustice et al. 2005], achieve exactly sparse formulations and hence the above approximations are no longer necessary.

The drawback of this technique is that the world representation is encrypted inside the information vector and matrix. The translation into a more geometrically sound representation requires a full-size matrix inversion and hence the constant-time properties are lost. However, this translation is not a part of the filter loop (it is actually analogous to the 'estimation' steps we perform on filtering, which do not affect the computation of the posterior —Chapter 3) and can eventually be performed at a much lower rate.

4.3.2 FastSLAM2.0

FastSLAM2.0 [Montemerlo et al. 2003] boards the complexity problem from a radically different perspective. Now, the Particle Filter is used in a Rao-Blackwellized form to divide the SLAM state vector (*i.e.* robot state and landmarks positions) into two differentiated parts. One part, the robot state vector, shows the main non-linearity problems and its *pdf* is approximated by a set of particles. The other part, the landmarks positions, are as usual modeled by Gaussian *pdfs*. As it is the norm in SLAM, the links between the robot and the landmarks arise exclusively from robot-centered measurements. This, when the robot's *pdf* is a particle, permits a complete de-correlation of the robot-landmarks set with the result of one robot pose (one particle) and a set of independent landmark Gaussian estimates. For each robot particle, a set of EKF is set, one per landmark, which exhibit constant update time just by limiting the number of simultaneous landmark measurements. The whole SLAM *pdf* is hence a set of particles and a set of sets of EKF. With a special and accurate treatment of the different

[1] The Extended Information Filter is the dual formulation of the EKF in which the couple *mean vector* and *covariances matrix* is substituted by the couple *information vector* and *information matrix*. The information matrix is just the inverse of the covariances matrix.

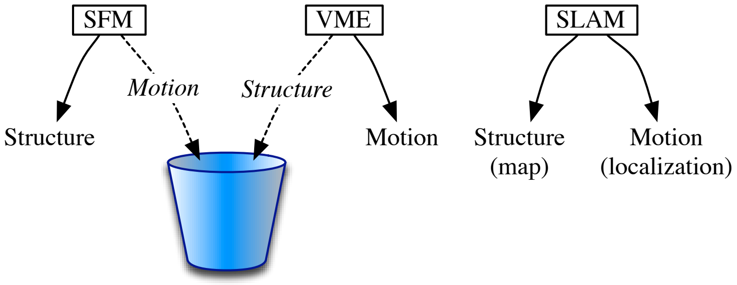

Figure 4.4: SFM and VME compared to SLAM. To perform SLAM, one could almost take the bin and run!

operations inside the filtering loop, all time steps and measurement steps for the whole SLAM system can now be computed in constant time if we just limit the number of particles and the number of simultaneous observations.

4.4 Alternatives to incremental formulations

When the exteroceptive sensors are video cameras, we may also consider two other techniques that relate very closely to SLAM: Visual Motion Estimation (VME) and Structure From Motion (SFM).

VME is conceived to obtain the robot ego-motion from the study of the series of images that their cameras take [Mallet et al. 2000]. Visual features are matched across pairs of stereo images taken at regular time intervals. An iterative minimization algorithm is run to recover the stereo rig motion, which is then transformed into robot motion. For this, the algorithm needs to recover some structure of the 3D points that correspond to the matched features[2], but this information is not exploited for other tasks and is therefore discarded. The system must work in real time as the robot localization is needed online.

SFM has the opposite objective: recover objects 3D structure from images taken from different points of view. This time with just a single camera, a similar iterative minimization scheme called Bundle Adjustment (BA) is responsible of obtaining, up to an unknown scale factor, the structure of the 3D points corresponding to the matched features. For that, the algorithm needs to recover all the positions from where the pictures where taken, which define the camera trajectory. This information is not exploited for other tasks and is therefore discarded. The system is normally run off-line as the interest is in recovering the structure of objects with potentially a very high precision and a huge number of points.

Basically, both techniques are solving the same problem that SLAM does. With the advent of today's fast computers and the apparition of software libraries that efficiently implement the BA algorithm [Lourakis and Argyros 2004], several authors have used it in real-time to perform SLAM. A deeper study showing the links between SLAM and BA has been published short ago [Konolige 2005].

[2] Although using the properties of the Fundamental matrix this structure could be actually hidden.

Part II

Towards vision based SLAM with moving objects tracking

Chapter 5

Probabilistic vision

> I wonder whether the quality of our comprehension depends on the dimensionality of the message. I mean: a text or a spoken discourse, for instance, is one dimensional, is like a line in time, a sequence of transitory stimuli, which are lost in the past and thus require an effort of memorization. In an image, a sketch for instance, which is two dimensional, time does not intervene. A video sequence would then be three dimensional, four dimensional with the addition of word, again with the discursive role of time. I wonder whether the power of image is not precisely this: the fact of being two dimensional and timeless.

5.1 Introduction

Images: *two dimensional* so that we are free to explore them in the directions we prefer; but *timeless* over all, so that it is again us, the receptors, who decide where to concentrate more attention, and when to revisit already perceived fragments. The reflection is not done because we are dealing with vision —not at all actually. It is done because in this chapter we will make an intensive use of images. Because, in this short chapter, we will only give intuitions, concepts, ideas. In a way, this chapter is the union of Geometry with Probabilities applied to Vision. One way to geometrically reason, from the visual evidence, about the external world.

For this, we deliberately choose a geometrical language to cover the following topics of probabilistic nature:

1. How to visualize a probability distribution function of the position of certain points in the 3D space (what we call *landmarks*) from the evidence extracted from their projections in the image (what we call *features*). We will introduce for this the concept of *conic ray.*

2. How to visualize the *composition* of the conic ray with the two main operations of filtering: the smoothing effect of the prediction convolution (when the robot moves for instance) and the refinement effect of the correction product (when the information from another image of the landmark wants to be incorporated).

3. How to determine under which conditions a set of observations of this landmark, from disparate points of view which are themselves not precisely specified, permits us to say that the landmark is confined within a closed and relatively small region in space. We will talk of *3D observability.*

4. How to make use of all the knowledge we have about the system to guide and render more robust the feature matching algorithms. We will refer to *active feature search*, something already used by other authors.

5.2 The conic ray

The conic ray is just the geometrical conceptualization of the knowledge that the position $\mathbf{y}$ in the image of a newly detected feature provides about the position $\mathbf{x}$ in the 3D world of the landmark that produced it. This knowledge, as it has been introduced in the Filtering chapter, is described in probabilities by the conditional density $p(\mathbf{y}_k|\mathbf{x}_k)$ appearing in the correction equation (3.17). At first time observation we can set $k = 0$ and the mentioned equation reduces to the Bayes rule:

$$p(\mathbf{x}_0|\mathbf{y}_0) = \frac{p(\mathbf{x}_0) \cdot p(\mathbf{y}_0|\mathbf{x}_0)}{p(\mathbf{y}_0)}. \tag{5.1}$$

where $p(\mathbf{y}_0) = \int p(\mathbf{x}_0)\, p(\mathbf{y}_0|\mathbf{x}_0)\, d\mathbf{x}_0$ is independent of $\mathbf{x}_0$. Notice that in the general case we do not have any previous knowledge on the position of the landmark and hence $p(\mathbf{x}_0)$ is uniform. Considering an infinitely large image plane[1] and no previous knowledge about the measurements, $p(\mathbf{y}_0)$ is also uniform. This absolute lack of previous knowledge leads to

$$p(\mathbf{x}_0|\mathbf{y}_0) = p(\mathbf{y}_0|\mathbf{x}_0) \tag{5.2}$$

which is a pure inversion of conditionings.

The camera provides the knowledge $p(\mathbf{y}_0|\mathbf{x}_0)$, that is to say it gives us a measurement $\mathbf{y}_0$ from a point in space $\mathbf{x}_0$. We are interested in visualizing the knowledge $p(\mathbf{x}_0|\mathbf{y}_0)$ in order to gain intuition on the way we can infer information about the 3D world from the information on the 2D images. For this, let us neglect by now the index $k = 0$ indicating time and consider, in a simplified 2D world, a normalized camera that projects 2D points into 1D image points (Chapter 2). With a convenient adoption of the XZ axes aligned as $\mathcal{S}\{UF; U\}$ we have that a point $\mathbf{x} = (x, z)$ in this sensor frame projects into the point $\mathbf{u} = u$ as:

$$u = \frac{x}{z}.$$

This projected point is then measured. The measurement contains imprecisions that are modeled by an additive Gaussian noise as follows:

$$y = u + \epsilon$$

with $p(\epsilon) = \mathcal{N}(\epsilon; R)$. By doing $\epsilon = y - u$ we can rewrite the *pdf* $p(\epsilon)$ (as in Remark 3.4) as follows

$$p(y|\mathbf{x}) = \mathcal{N}\big(y - \frac{x}{z}, R\big) \tag{5.3}$$

which from (5.2) and a little touching-up leads also to

$$p(\mathbf{x}|y) = \mathcal{N}\big(\frac{x}{z} - y, R\big) \tag{5.4}$$

which are both the same function of $\mathbf{x}$ and y. Now consider these two complementary cases:

[1] This assumption permits to easy up the discourse and is quite irrelevant in practice: out of the image plane limits there is no measurement and hence no necessity to describe anything.

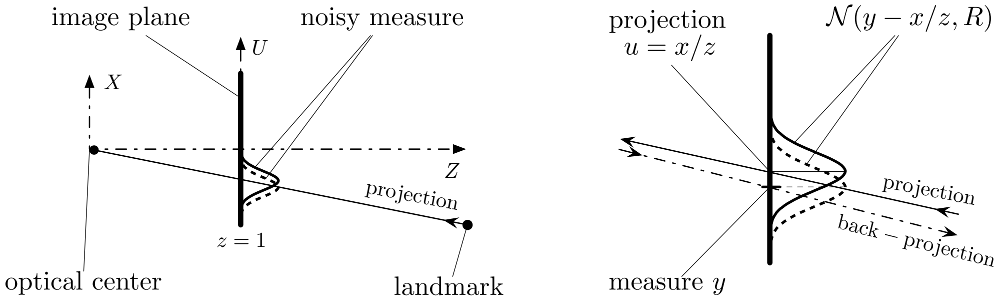

Figure 5.1: Projection and back-projection operations in the presence of Gaussian noise. On the right, a detail of the image plane, close to the detected pixel, showing the effect of the measurement error separating both projection and back-projection axes.

- *Projection operation.* As a function of $y \in \mathbb{R}$, *i.e.* when $\mathbf{x}$ is known, the density (5.3) is a Gaussian in the image plane centered at the true projection $u = x/z$ (solid lines in Fig. 5.1). This is the real-world operation performed by the camera: from points $\mathbf{x}$ to images y.

- *Back-projection operation.* As a function of $\mathbf{x} \in \mathbb{R}^2$, *i.e.* when y is known, the density (5.4) defines a 'manifold' in $\mathbb{R}^2$. The intersection of this 'manifold' with the image plane $z = 1$ is the Gaussian curve $p(u) = \mathcal{N}(u - y, R)$ of variance R which is centered at the measurement y. (Notice that this second Gaussian –dotted lines in Fig. 5.1– is a replica of the first one, but their means are not coincident.) This is the operation the robot's brain must perform: from images y to points in space $\mathbf{x}$. This is what we are interested in.

The manifold produced by the back-projection operation is shown in Fig. 5.2. The difference between the projection and the back-projection axes is clearly visible: *a)* the landmark $\mathbf{x}$ is projected onto the image; *b)* the measurement y of this projection is noisy and does not coincide with the true value; hence *c)* the back-projected axis is not going to pass exactly through the landmark $\mathbf{x}$. However, from the fact of having measured y, equation (5.4) says that the probability of finding the landmark somewhere in the XZ plane is high for those points *close* to the back-projected axis. This distance to the axis is measured via the number of standard deviations $\sigma = \sqrt{R}$, which define iso-probable surfaces (limited by the labeled lines 1σ, 2σ, 3σ, etc.) that emanate from the optical center as shown.

The extension to the 3D case should be obvious. Let us go then to 3D, but let us change a little bit the discourse to adopt a more geometrical approach as follows. Consider a feature is detected in a certain point of the image. This point, as we know, is a noisy measurement of the true projection. Consider this noise to be Gaussian and draw its *pdf*'s iso-probable curves, the n-sigma bound regions, which define concentric elliptic regions of confidence.[2] This permits to say, with a certain language abuse, that the *pdf* of the landmark's projection is an elliptic planar region centered at the detected pixel. When any of these ellipses is back-projected it produces a conic volume in 3D space: the *conic ray*, which extends to infinity. Fig. 5.3 shows the 3σ conic ray.

The conic ray defines the *pdf* of the landmark position in the following sense: the landmark is inside the conic ray with the same probability (99% in the 3σ example of the figure) that

[2] See Appendix B for more details on the ellipsoidal representation of Gaussian *pdfs*.

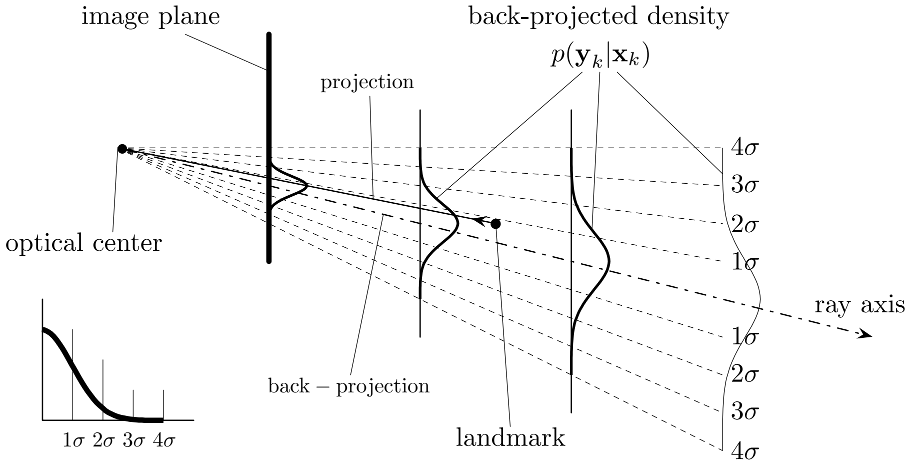

Figure 5.2: Manifold of probability in the 2D space induced by a noisy measurement of the projection of a point in space. The conic ray is just the extension of this *pdf* to the 3D case.

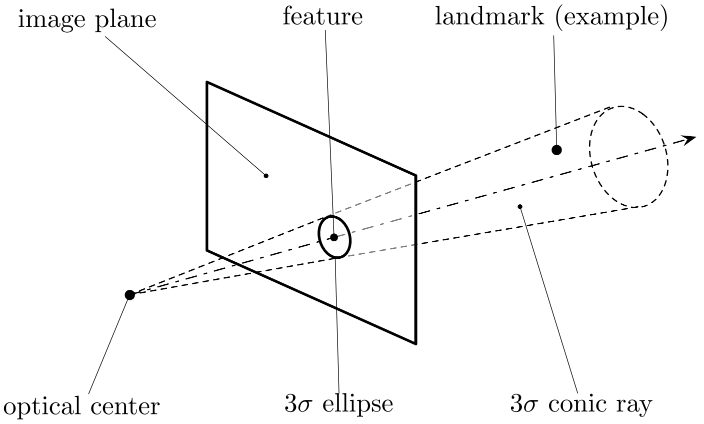

Figure 5.3: The conic ray extends to infinity.

its projection was in the 2D ellipse. The landmark is more likely to be close to the ray axis, but in this axial direction the conic ray gives no information: we have the same result as in 2.3.3 in Chapter 2 which said '*the depth of a 3D point is not observable from a single picture shot*', but we have added a mathematically tractable object, the conic ray, which has been mathematically and geometrically described.

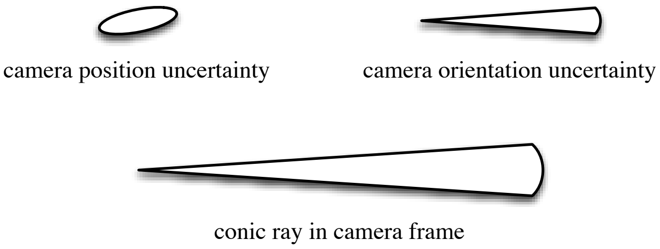
camera position uncertainty
camera orientation uncertainty
conic ray in camera frame

Figure 5.4: Geometric interpretation of the *pdfs* in our system.

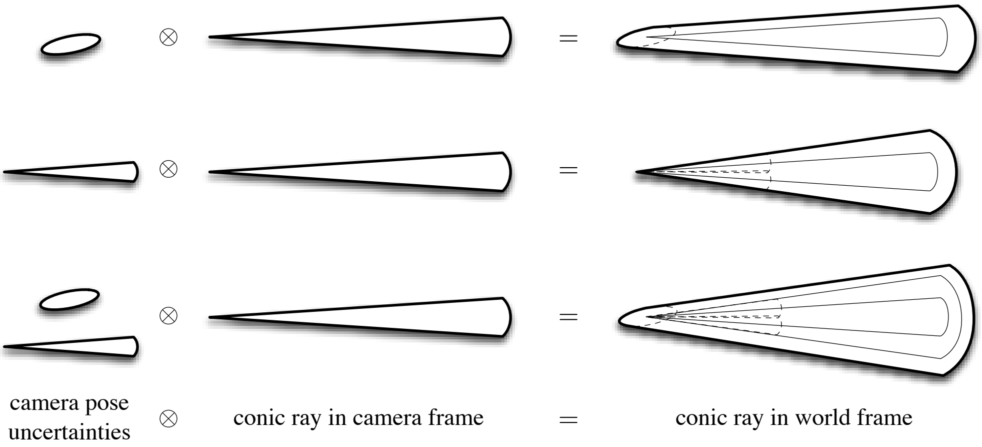

Figure 5.5: Convolution-like compositions of the camera pose uncertainty with the conic ray. The knowledge on the position of a landmark when the camera pose is uncertain is less accurate. The camera position uncertainty produces rays with a rounded tip but with the same angular aperture (*top*). The uncertainty on the camera orientation traduces to rays with greater angular aperture (*center*). The general case is a combination of both (*bottom*).

5.3 Knowledge composition

5.3.1 Motion: Convolution-like composition

We have seen the conic ray defined in the camera frame. When this camera frame is itself uncertain it is intuitively sound to affirm that this conic ray, when expressed out of this camera frame (for example in the world frame), will be less accurate.

A purely geometrical interpretation follows: consider a camera precisely situated in the origin of the world coordinates, with absolutely no uncertainty. A conic ray in the world frame would be exactly that in the camera frame which we have just seen. Apply now one step of uncertain movement to this camera. The resulting camera pose (the camera extrinsic parameters) will have both position and orientation uncertainties. Consider these uncertainties to be Gaussian and represent them geometrically by the shapes in Fig. 5.4: a 3σ ellipsoid for the position uncertainty and a 3σ cone for the orientation uncertainty.

If at this point we detect a feature in the image, we know from Chapter 3 that the resulting *pdf* (the conic ray in world frame) will be a convolution-like operation of the camera uncertainties with the conic ray in camera frame. This is illustrated in Fig. 5.5. The rays get

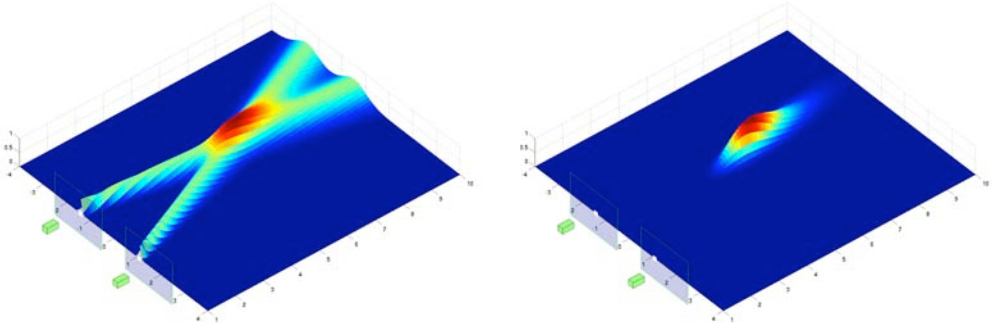

Figure 5.6: A point-to-point multiplication of two conic rays. *Left:* before multiplication. *Right:* after. This product defines the *pdf* of the landmark position as the intersection of rays. In this case the intersection produces a well defined, closed region, which means that the landmark can be considered fully 3D observed.

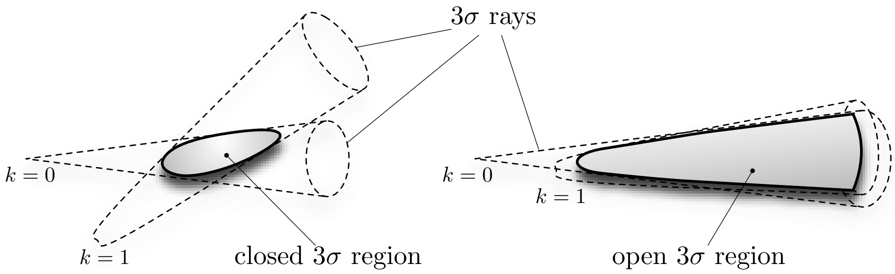

Figure 5.7: Two intersecting conic rays may define a closed 3D region. The landmark is inside with known probability. They may also define an open region.

less accurate when the camera that produced them is not accurately positioned.[3]

5.3.2 Observation: Product-like composition

We are interested now in the correction step of filtering, that is how to refine the knowledge we have on the position of the landmark by incorporating a new observation. As it has been seen, this refinement is done by multiplying the *pdf* of our knowledge by the *pdf* of the observation. For the case of a single landmark this is just the product of conic rays defined from different viewpoints, where the specification of one viewpoint with respect to the other one may be uncertain as we have just seen. A simple example representing a stereo observation is shown in Fig. 5.6.

This way the conic ray can help us recovering the landmark's depth: under certain conditions, the product of intersecting rays will define a closed region inside which the 3D point will be confined with a known probability (Fig. 5.7). These conditions are studied in the next section.

[3] Notice that the intervening uncertainties are usually independent and thus they add-up in an orthogonal way, *i.e.* it is the covariances that are added, not the standard deviations. This should not worry the reader as the EKF-SLAM formalism, which we will use to mathematically perform all these operations, naturally accounts for all system correlations.

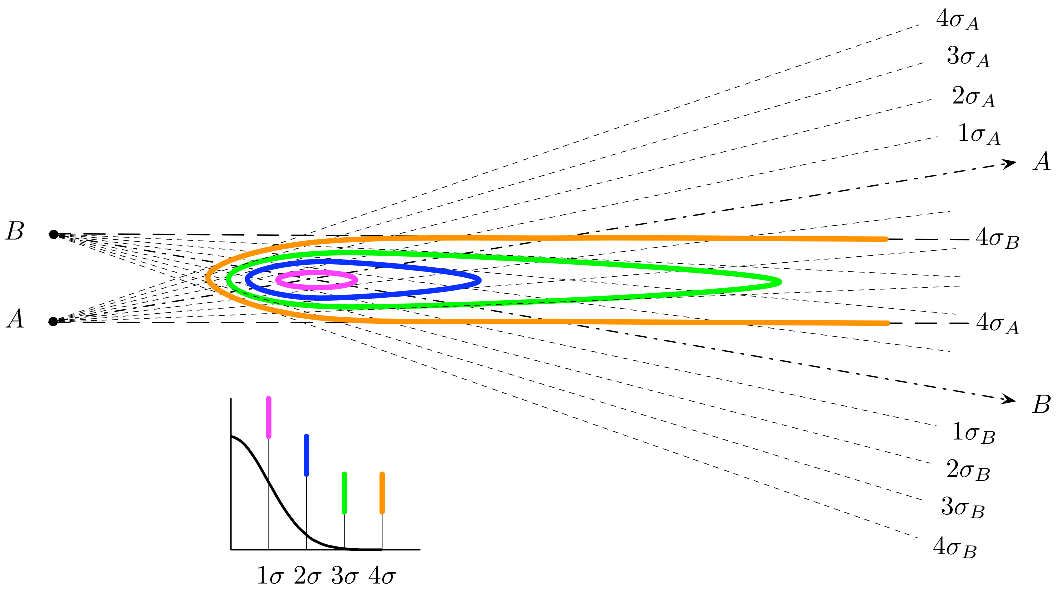

Figure 5.8: Different regions of intersection for 4σ (orange), 3σ (green) and 2σ (blue) ray widths (in 2D). When the two external 4σ bounds are parallel the 3σ bound defines a closed region. The angle between rays axes A and B is $\alpha = 4\,(\sigma_A + \sigma_B)$.

5.4 3D observability from vision

Just a couple of ideas (not to be strictly interpreted) to help to understand the observability concept that we use (Figs. 5.7 to 5.10), and to define the conditions under which a landmark will be considered fully observed. Let us consider two features extracted from two images and matched because they correspond to the same landmark. Their back-projections are two conic rays that extend to infinity. The angular widths of these rays are defined by concentric ellipses parametrized by the standard deviations σ_A and σ_B of the angular uncertainties, which depend on the camera angular resolution (camera intrinsic parameters), on the camera pose precision (extrinsic parameters) and on the accuracy of the feature detecting and matching algorithms. We say that the landmark's depth is observed if the region of intersection of these rays is *a)* closed and *b)* sufficiently small. If we impose, for example, the two external 4σ bounds of the rays to be parallel (Fig. 5.8), then we insure that the 3σ intersection region (which covers 97% probability) is closed and that the 2σ one (covering 74%) is small. The angle α between the two rays axes is then $\alpha = 4(\sigma_A + \sigma_B)$.

In 2D, we can plot the locus of those points where two angular observations differ exactly in this angle α. In the case where σ_A and σ_B can be considered constant α is constant too and the locus is then circular (Fig. 5.9). Depth is observable inside this circle; outside it is not. The circle's radius is directly proportional to the distance d between the two cameras. In 3D, the observability region is obtained by revolution of this circle around the axis joining both cameras, producing something like a torus-shaped region.

In the general case angular uncertainties σ_A and σ_B are not uniform in all directions and thus the angular aperture α neither. This means that these observability regions are not exactly circular, not exactly torus-shaped, etc. But the main conclusions still hold: *a)* the maximum full observability range is in the direction perpendicular to the axis joining both camera positions; *b)* this range is proportional to the distance between both cameras; and

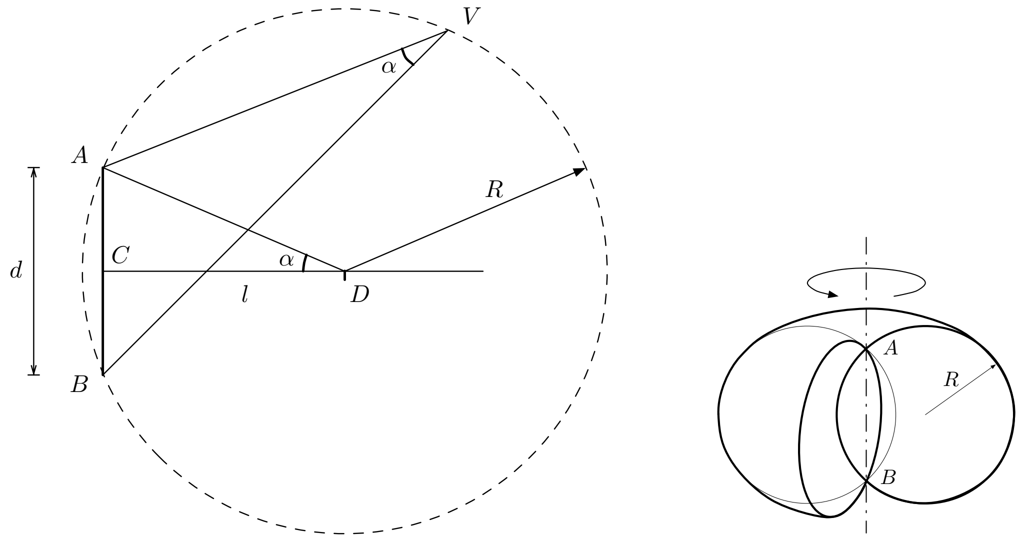

Figure 5.9: Locus of points at constant angular aperture from points A and B. *Left:* Take $C = (A+B)/2$ and trace the line l through C perpendicular to AB. In this line, mark the point D so that $\widehat{ADC} = \alpha$, the desired aperture angle. Use D as the center of a circumference of radius $R = \|A - D\|$ —this radius is obviously proportional to $d = \|A - B\|$. The circumference goes over A and B, and every point V on it precisely satisfies $\widehat{AVB} = \alpha$. Any point U inside it satisfies $\widehat{AUB} > \alpha$, while points W outside get $\widehat{AWB} < \alpha$. *Right:* In 3D, the locus surface is obtained by revolution of this circumference around the AB axis, producing a torus shape with a degenerated central hole.

c) depth observability is extremely weak for those points lying close to this axis.

In a stereo configuration or for a lateral motion of a moving camera like in aerial images the observability region is in front of the sensor (Fig. 5.10 *left*). This confirms the intuition that with a single camera the best way to recover 3D information is by performing lateral motions, as the change of viewpoint amplitude will be maximum.

Consider at the opposite extreme the specially difficult case of singular motion, *i.e.* a single camera moving forward (Fig. 5.10 *right*): in the motion axis depth recovery is simply impossible. Close to and around this axis, which in robotics applications is typically the zone we are interested in, observability is only possible if the region's radius becomes very large. This implies the necessity of very large displacements d of the camera during the landmark initialization process. As we will see in vision-based SLAM, this will be possible only by using what we will call *undelayed initialization methods*, which initialize the landmark in the SLAM map at the first observation, before it is fully 3D observed.

5.5 Active feature search

We regard now the problem the other way around. From the knowledge we have about the landmarks in the 3D world and the camera pose, we want to infer the characteristics of their projected features. These *expected* characteristics will then be exploited to accelerate the feature matching procedures which are normally very time-consuming. The aim of *active feature search* is the economy of computing resources by using very simple appearance-based feature descriptors together with very simple correlation-based matching, which are guided by the knowledge we have been able to collect on the system. Fig. 5.11 illustrates the required

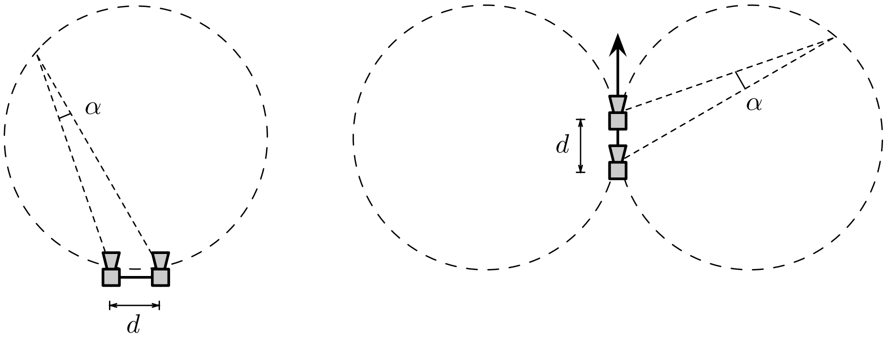

Figure 5.10: Simplified depth observability regions in a stereo head (*left*) and a camera traveling forward (*right*). The angle α is the one that insures full 3D observability via difference of points of view.

steps and shows where and how this knowledge is generated and exploited.

The information we can infer on the appearance the feature is likely to have is basically, but not limited to, the following:

1. The *search region* in the image where the feature must be found with known probability. This is done by simply projecting on the image plane the region of confidence of the landmark position (Fig. 5.12).

2. The *scale factor* the feature appearance has suffered from the initial perception, when the descriptor was set, to the current one. This factor is just the ratio between the current landmark depth and the depth at the initial camera pose. These depths are estimated from the initial camera pose, which must be stored as part of the landmark descriptor, and the current estimates of landmark position and camera pose.

3. The *rotation angle*. This is just the opposite of the camera rotation around the optical axis (*i.e.* the *roll* Euler angle) from the first observation to the current one.

4. Other higher-order transformations (perspective, projective, etc.). These operations normally require an incremental enrichment of the feature descriptor,[4] for example the determination of the vector which is normal to the landmark's surface. They are not considered in this work.

[4] Operations *1)* to *3)* perform translation, scaling and rotation of the feature appearance-based descriptor, which are linear 2D operations. Perspective and projective operations need additional 3D considerations which require richer, 3D information of the landmark. This 3D information is obviously not observable from a single picture shot.

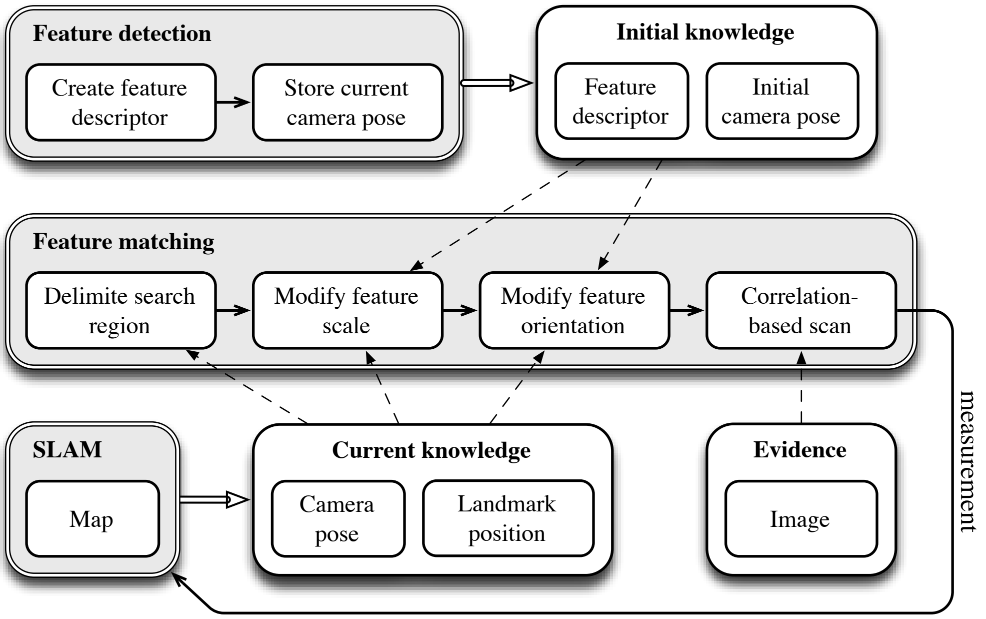

Figure 5.11: Active feature search. Both initial knowledge collected at the time of feature detection and current knowledge contained in a SLAM map are exploited to guide the feature matching procedure.

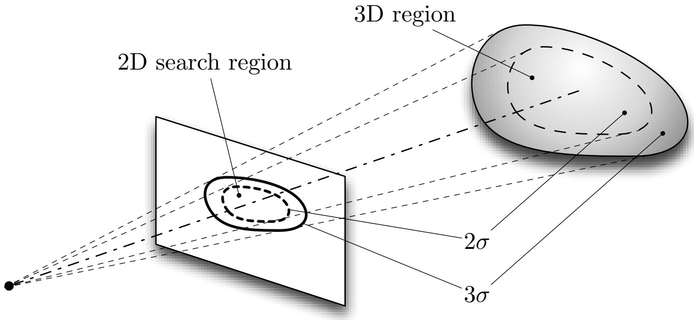

Figure 5.12: Determining the search region. A simple projection of the landmark's confidence region defines another confidence region in the image plane which is used as the feature's search region. Using 2σ bounds we get 87% confidence. The 3σ bound gives 99%. Higher $n\sigma$ are not justified because the region's area growth (which is $O(n^2)$) is not accompanied by a confidence increase.

Chapter 6

Mono-camera SLAM

6.1 Introduction

Solving the Simultaneous Localization and Mapping problem (SLAM) by means of Vision sensors is a relatively new experience. We pointed in the SLAM chapter that the estimation side of SLAM was reaching saturation and that new efforts are being devoted to the perception side. In this trend, Vision is a specially appealing choice: cameras are cheap, small and reliable, and are capable of providing a huge amount of spacial information.

In order to directly use the available SLAM methods, a pair of cameras in *stereo* configuration can be set up to provide range and bearing information of the surrounding world. The region where this full 3D information can be extracted from the stereo image pairs has been studied in the previous chapter: in front of the stereo sensor we dispose of an approximatively circular region whose dimensions depend basically on the stereo base-line (the distance between both cameras) and the angular accuracy of the cameras and image processing algorithms. The *advantage* of a stereo solution is therefore its capability of providing an invertible 3D observation function and hence to naturally allow for immediate or *undelayed* landmark initializations. The *drawbacks* of stereo are considered here as follows:

1. *Limited 3D observability range.* While a camera is capable of sensing visible objects that are potentially at infinity (consider taking a picture of the pole star if you doubt of this assertion), having to work exclusively with information from a limited range is simply a technological pity. It is like, having our two eyes, we were obliged to neglect everything farther than a certain range, what I call walking inside dense fog: it is just much easier to get disoriented, to loose the spacial references and, in a word, to get lost.[1] It is therefore a bad starting point to SLAM —or a sad one if you want.

2. *Mechanical fragility.* In order to increase the 3D observability range we need to simultaneously increase the base line and keep or improve the sensor precision. This is obviously a contradiction: larger assemblies are less precise when using the same mechanical solutions; to maintain accuracy with a larger assembly we must use tougher structures, that will be either heavier or more expensive, if not both. Alternatively, self-calibration procedures could be used, something that is not 'included' in the classical SLAM solutions.[2]

[1] Further consider what could have been the history of northern ocean navigation without the pole star.

[2] For the reasons that will be explained, self-calibration of a stereo rig is explored in the next chapter.

These drawbacks are eliminated with the use of a single camera, with the additional gain in cost, weight, spacial compactness, robustness and hardware simplicity, and at the cost of losing one dimension of the world we want to observe: a camera cannot provide the distance to the perceived objects. If we want these benefits, we must overcome their cost by providing additional intelligence: using such a sensor obliges us to develop more ‘intelligent’ methods. These methods for the SLAM problem receive the generic name of *Bearings-only SLAM*, or *mono-vision SLAM* in our case of vision.

This chapter is devoted to elucidate several implications of the adoption of a bearings-only solution to SLAM, and we will do it in the general case which will force us to consider the worst-case scenarios, those that provide the poorest observability conditions. We will adopt for this the EKF-SLAM framework (Chapter 4) as it is easy to set-up and because, at the time of starting our works in 2003, it was the only one to have proven its feasibility to perform real-time, visual-based SLAM at the video frequency of 30Hz [Davison 2003].

The chapter also covers the perception aspects that are necessary to close the filtering recursive loop of SLAM. As introduced in Chapter 5, the active feature search approach is adopted. This approach has been successfully demonstrated in the most advanced real-time vision-based algorithms, notably the already mentioned works [Davison 2003; Eade and Drummond 2006] as it allows for robust operation with a considerable economy of computer resources. This is due to the fact that, knowing where in the image the information must be extracted, the image processing (*i.e.* the set of algorithms responsible to translate luminosity information into geometric information), which is very time-consuming, can be limited to a tiny percentage of the image surface.

What this chapter does not pretend (and the rest of this thesis neither) is to build a complete SLAM system to compete against the numerous and performing proposals that are rapidly appearing in the last few years [Lisien et al. 2003; Thrun et al. 2004; Montemerlo et al. 2003]. In particular, we are not addressing the problem of large area mapping or large loop closings that are often the motivations of many of such mentioned works. For a comprehensive survey on the state-of-the-art in SLAM we recommend reading [Durrant-Whyte and Bailey 2006; Bailey and Durrant-Whyte 2006] and selected references therein.

This chapter, which is the heart of this thesis and the longest one, is organized as follows. In Section 6.2 the distinction between delayed- and undelayed- landmark initialization families of methods is explained and the benefits of the undelayed ones are highlighted. In Section 6.3 we present the main difficulties we will encounter when dealing with undelayed initialization with bearings-only measurements. Section 6.4 develops the principal contribution of this thesis to the problem solution. Section 6.5 covers in detail the visual perception methods. Finally, experimental results are exposed in Section 6.6.

The research contained in this chapter led to the following publications: [Solà et al. 2005; Solà 2005; Lemaire et al. 2005b; Lemaire et al. 2005a; Lacroix et al. 2005].

6.2 Delayed versus undelayed initializations in Bearings-Only SLAM

Landmarks Initialization in Bearings-Only EKF-SLAM is a delicate task. EKF requires Gaussian representations for all the involved random variables that form the map (the robot pose and all landmark’s positions). Moreover, their variances need to be small to be able to properly approximate all the non-linear functions with their linearized forms. From one bearing

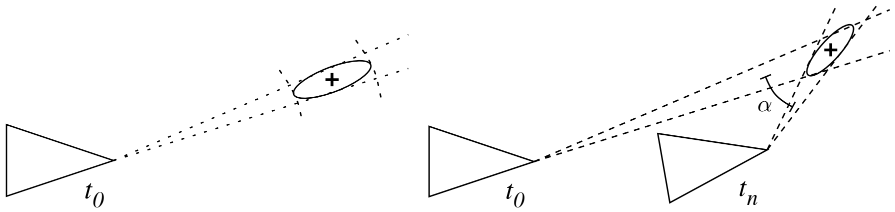

Figure 6.1: Landmark initializations. *Left*: Range and Bearing SLAM. *Right*: Acquiring angular aperture α in Bearings-Only SLAM.

measurement, we cannot establish an estimate of the landmark position that satisfies this fundamental rule. This estimation is only possible via successive measures from different points of view, when enough angular aperture has been accumulated (Fig. 6.1).

This reasoning leads to systems that have to *wait* for this angular aperture to be available before initializing the landmark in the SLAM map: what we name the *delayed methods*. Davison [2003] uses a separate Particle Filter to estimate the distance. Initialization is deferred until the range variance is small enough to consider a Gaussian estimate. In [Bailey 2003] past poses of the robot are stacked in the map, with their associated bearings stored apart, until the landmark *pdf* is considered sufficiently close to a Gaussian shape to permit a Gaussian initialization. This is evaluated by means of a numerical implementation of the Kullback distance [Kullback 1959] which is computationally expensive. Once initialized, the set of observations from all stored poses is used in a batch update to refine and correct the whole map. These methods suffer from two drawbacks: they need a criterion to decide whether or not the angular aperture is enough, and they introduce a delay in the landmark initialization until this criterion is validated. This delay implies that the landmark is not mapped until its 3D position has been fully observed, therefore preventing the robot to use them for re-localization, which in certain situations (situations that have been avoided by the aforementioned works) can lead to important performance degradations. To keep this delay below reasonable limits one needs to assure, as we know now from Chapter 5, that the camera motion is not close to the direction of the landmark and that landmarks are not very far, that is to say, one needs to restrict operation to lateral camera motions in indoors environments.

Avoiding both criterion and delay is an interesting issue. The criterion is often expensive to calculate, and the system becomes more robust without it as no binary decisions have to be taken. Without the delay, having the bearing information of the landmark in the map permits its immediate use as an angular reference. We speak then of *undelayed methods*: those methods that include the landmark information in the map from the first observation, *i.e.* before the landmark has been fully 3D observed. This allows us to adequately use the information provided by the observation of landmarks that lie close to the direction of motion of the robot, or that are very far away, for which the angular aperture would take too long to grow. This is crucial in outdoor navigation where remote landmarks are everywhere, and where straight trajectories are common, with vision sensors naturally looking forward. This thesis is specially dedicated to provide satisfactory SLAM operation in such cases: notice that, besides the dense-fog effect of short-range SLAM, a frontally-blind mobile, which is incapable to consider those objects that are in front of his path, is very likely to crash (consider driving your car under the conditions of Fig. 6.2 *left*).

To our knowledge, only Kwok and Dissanayake [2004] propose an undelayed method, although the authors never remark it, and thus they cannot claim it as being something partic-

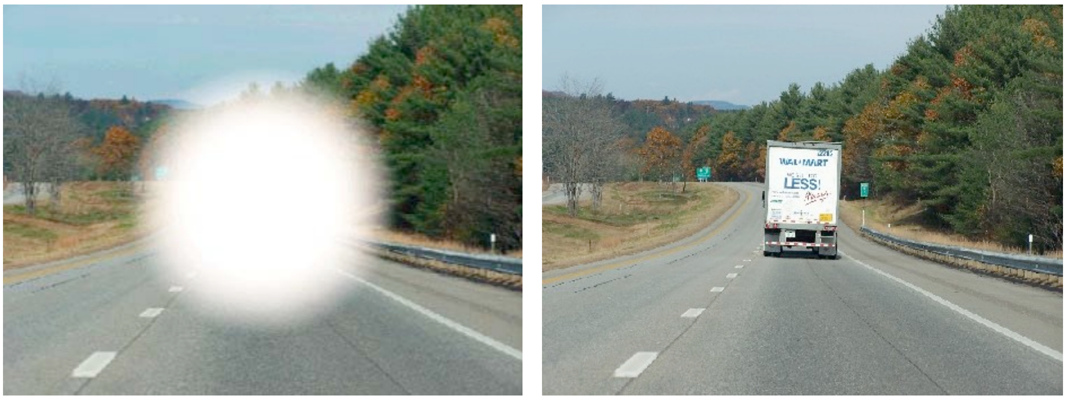

Figure 6.2: Delayed and undelayed operations. Frontal-blindness effect of delayed initialization (*left*) which leads to SLAM systems that only take into account fully observed landmarks, thus not those close to the motion axis. By considering also partially observed landmarks, an undelayed operation (*right*) is able to exploit all the available information.

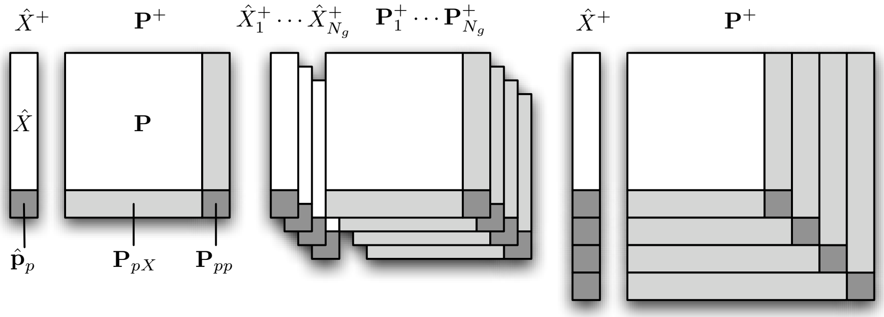

Figure 6.3: Landmark initializations in EKF-based SLAM systems. Mean and covariances matrices are shown for *Left*: EKF-SLAM; *Center*: GSF-SLAM; *Right*: FIS-SLAM.

ularly interesting. They define a set of Gaussian hypotheses for the position of the landmark, and include them all inside the same EKF-SLAM map from the beginning. On successive observations, the sequential probabilities ratio test (SPRT) based on likelihood is used to prune bad hypotheses, and the one with maximum likelihood is used to correct the map. The way these hypotheses are set up is not detailed, and convergence and consistency issues are not discussed. In some key aspects, our approach is surprisingly similar to this one.

The distinction between delayed and undelayed initialization methods has been, for the first time in the SLAM literature, highlighted by the author in [Solà et al. 2005]. In this paper we stated the benefits of undelayed initializations over delayed ones, detected the associated difficulties, and gave two EKF-based solutions which rely on a special multi-Gaussian approximation of the conic ray: *a)* the theoretically motivated one, based on the Gaussian Sum filter (GSF), cannot run in real time because of a combinatorial explosion of the problem complexity (Fig. 6.3 *center*); and *b)* an approximation to the GSF, based on the proposed new technique named *Federated Information Sharing* (FIS), which is well suited for real-time operation.[3] This chapter follows and extends the discourse of the paper.

[3] To our surprise, this paper [Kwok et al. 2005] from the same authors of [Kwok and Dissanayake 2004] was being published at the same time of ours. It defends GSF-SLAM, precisely what we want to avoid.

Our proposed FIS method is an approximation of the GSF that permits undelayed initialization with a simply additive growth of the problem size (Fig. 6.3 *right*). At the first observation, the robot only knows the optical ray on which the landmark is located. This ray, with associated covariances, defines a conic probability density function (*pdf*) for its position: the conic ray, which we already know from Chapter 5. A minimal representation of this *pdf* is introduced as a geometric series of Gaussians. The terms of this series are all included in one single EKF-SLAM map, as if they corresponded to different landmarks. As it is usual with approximations, this representation increases the EKF inherent risks of *inconsistency* and *divergence*, which we discuss. To minimize these risks we propose a strategy for all the subsequent updates which we name Federated Information Sharing (FIS). We define a very simple criterion for pruning the less likely members of the ray.

Some works have more recently appeared that surpass the performances of the ones proposed here. Nevertheless, all the concepts exposed here, which as stated appeared in our work [Solà et al. 2005], are taken into account and notably contribute to the good performances of these newer solutions. The FastSLAM2.0 algorithm is used by Eade and Drummond [2006], with an inverse parametrization of the unmeasured depth and the use of the epipolar constraint to improve camera localization when observing partially observed landmarks (*i.e.* remote landmarks or those lying close to the motion axis). Montiel et al. [2006] use the same inverse-depth parametrization in the EKF-SLAM framework leading to similar results in undelayed operation. The first proposal clearly surpass the second one and ours in the capability of producing larger maps thanks to the constant-time complexity of the FastSLAM2.0 algorithm. However, some doubts still exist on whether FastSLAM is able to keep track of all system correlations as EKF-SLAM does.

6.3 Implications of an undelayed initialization in the EKF framework

In this section we show that the theoretically motivated solution to an undelayed initialization using a multi-Gaussian approximation for the landmark depth implies the abandonment of the EKF (Fig. 6.3 *left*) and that, following a multi-hypothesis reasoning, the proper way to include all the information in the map is the creation of a set of weighted maps, one for each hypothesis (Fig. 6.3 *center*). This will lead to intractable algorithms such as the Gaussian Sum Filter (GSF) [Alspach and Sorenson 1972], in which the computational load grows multiplicatively with the number of hypotheses and exponentially with the number of simultaneous initializations.

6.3.1 Unmeasured depth and initialization in EKF

Landmark initialization in EKF-SLAM was naturally undelayed because of the availability of Range-and-Bearing measurements as we saw in Chapter 4. There, we defined the variable change $\mathbf{w} = \mathbf{h}(\mathcal{R}, \mathbf{p})$, from robot pose $\mathcal{R}$ and landmark position $\mathbf{p}$ to the range-and-bearing representation $\mathbf{w}$, which led to the noisy measurement $\mathbf{y} = \mathbf{w} + \upsilon$, and inverted it to obtain $\mathbf{p} = \mathbf{g}(\mathcal{R}, \mathbf{w})$, from robot pose and range-and-bearing representation to landmark position. This was then linearized to obtain the Gaussian statistics of the augmented map.

In the Bearings-Only case the measurement is lacking the range information and the initialization procedure is not that straightforward. We separate range[4] s from bearing $\mathbf{b}$ and write

$$\mathbf{w} = \begin{bmatrix} \mathbf{b} \\ s \end{bmatrix} \tag{6.1}$$

so the measurement is now $\mathbf{y} = \mathbf{b} + \upsilon$. This leads to the re-definition of $\mathbf{g}(\cdot)$

$$\mathbf{p} = \mathbf{g}(\mathcal{R}, \mathbf{b}, s) \tag{6.2}$$

where all but range s can be safely considered Gaussian.

In the vast majority of cases, no –or little– a priori knowledge is available for the values of the unmeasured range s. Often we will be in the case where only a validity margin is known. In the case of an unmeasured depth, typical validity margin of s could be:

$$\begin{aligned} s &\in [0, \infty) \\ s &\in [0, s_{max}] \\ s &\in [s_{min}, \infty) \\ s &\in [s_{min}, s_{max}] \end{aligned}$$

Such validity margins clearly violate the Gaussian assumption (as they define uniform *pdfs*), and even in the case of reasonable reduction to Gaussian shapes, they will for sure violate the validity margins for the linearizations that are necessary in EKF. If for all other magnitudes in (6.2) the Gaussian and local linearity assumptions were reasonably valid hypotheses (as it is implicitly assumed in totally observable EKF-SLAM), we are now in a completely opposite situation.

Therefore, the landmark initialization method described in the EKF-SLAM section in Chapter 4 no longer holds. We have to stochastically re-characterize s and find a way to cope with its –very wide– probability density function $p(s)$.

6.3.2 On the two main stochastic approaches

Among the up to the date proposed solutions for $p(s)$ to address the initialization problem we find particle approximations, which lead to the implementation of dedicated Particle Filters, and Gaussian sum approximations, which lead to multi-hypothesis Gaussian Sum Filters. Let us write both representations, just as an illustration of how similar they are:

$$\begin{aligned} p_{particles}(s) &= \sum_{j=1}^{N_p} \rho_j \cdot \delta(s - s_j) \\ p_{Gaussians}(s) &= \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}(s - s_j; \sigma_j^2), \end{aligned}$$

where $\mathcal{N}(s - s_j; \sigma_j) = (\sqrt{2\pi} \cdot \sigma_j)^{-1} \exp(-(s - s_j)^2 / 2\sigma_j)$, and let us insist a little more and say that if we put the Dirac function as a Gaussian with zero variance:

$$\delta(s - s_j) \triangleq \lim_{\sigma_j \to 0} \mathcal{N}(s - s_j, \sigma_j^2)$$

[4] The unmeasured range is not necessarily an Euclidean distance. For vision it is more common to use depth.

then the similarities are astonishing.

Although Particle Filters are getting widely popular to solve a great variety of non-linear or non-Gaussian filtering problems, as it is our case, we must say that their usage is not always justified from the point of view of efficiency. In the case of slightly non-linear problems (such as bearing only measurements where local linearity can be always established) or mono-modal, continuous and largely derivable probability densities (as a uniform density would be except for its edges), techniques closer to the linear-Gaussian case should be encouraged, as this choice would go in favor of efficiency. In such cases the number of Gaussians N_g will be drastically smaller than the number of particles N_p. We will adopt a Gaussian mixture solution suited to our problem.[5]

6.3.3 Gaussian multi-hypothesized depth landmark initialization

Concise ways of getting good Gaussian mixture approximations for our application will be given later. By now, let us assume that we can get a good approximation of the probability density function of s in the form of a finite Gaussian sum:

$$p(s) \approx \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}(s - s_j; \sigma_j^2) \tag{6.3}$$

Landmark initialization consists of stacking the new landmark position $\mathbf{p}$ into the map $X^\top = [\mathcal{R}^\top\ \mathcal{M}^\top]$ as

$$X^+ = \begin{bmatrix} X \\ \mathbf{p} \end{bmatrix}$$

and finding its *pdf* conditioned to the bearing observation $\mathbf{y} = \mathbf{b} + v$. This can be done as follows. Consider depth s (unobservable) and map X of being independent of $\mathbf{y}$ and start by writing the conditional densities of all the knowledge we have: map, bearing and depth:

$$\begin{aligned} p(X|\mathbf{y}) &= p(X) = \mathcal{N}(X - \bar{X}; \mathbf{P}) \\ p(\mathbf{b}|\mathbf{y}) &= \mathcal{N}(\mathbf{b} - \mathbf{y}; \mathbf{R}) \\ p(s|\mathbf{y}) &= p(s) = \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}(s - s_j; \sigma_j^2). \end{aligned}$$

Further consider X, $\mathbf{b}$ and s of being mutually independent and write their joint conditional

[5] But why did Particle Filters, invented in the very last eighties and developed all along the nineties, take such a predominant role in front of the Gaussian mixture solutions, which date back from the early seventies? This curious fact has nothing to do with the fall of the hippie movement in the mid-seventies, but rather in the great power and simplicity of the Particle Filter. As every particle can be understood as a deterministic entity (in the sense that it has null variance), intuition on what to do with it, and how, will help to rapidly set up a filter which will solve our problem. Also, this deterministic behavior of particles will allow us to make important simplifications (not approximations!) of the mathematical operations involved. And still, as the Particle Filter is a powerful and versatile tool to solve a great variety of filtering problems, even the most difficult ones, understanding this unique technique will put us in the position of solving any of such problems. Our word in the case we are dealing with is that these solutions will not be optimal in the sense of efficiency.

density as

$$\begin{aligned}
p(X,\mathbf{b},s|\mathbf{y}) &= \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}(X-\bar{X};\mathbf{P}) \cdot \mathcal{N}(\mathbf{b}-\mathbf{y};\mathbf{R}) \cdot \mathcal{N}(s-s_j;\sigma_j^2) \\
&= \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}\left(\begin{bmatrix} X-\bar{X} \\ \mathbf{b}-\mathbf{y} \\ s-s_j \end{bmatrix} ; \begin{bmatrix} \mathbf{P} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_j^2 \end{bmatrix} \right).
\end{aligned}$$

From the variable change (6.2) we can define this new one which applies to the whole system

$$\begin{bmatrix} X \\ \mathbf{p} \end{bmatrix} = \mathbf{g}'\left(\begin{bmatrix} X \\ \mathbf{b} \\ s \end{bmatrix} \right).$$

In order to map the *pdf* from $(X,\mathbf{b},s)$ to $(X,\mathbf{p})$ via $\mathbf{g}'(\cdot)$ we introduce the following proposition:

Proposition 6.1 (Variable change in *pdfs*). *Consider two n-dimensional random variables X and Y related by the invertible deterministic function $Y=f(X)$,* i.e. *$X=f^{-1}(Y)$. Both* pdfs *$p_X(x)$ and $p_Y(y)$ are related by the expression*

$$p_Y(y) = p_X(f^{-1}(y)) \left| \frac{\partial f}{\partial x^\top}\Big|_{f^{-1}(y)} \right|^{-1} \tag{6.4}$$

□

Considering now $\mathbf{P}$, $\mathbf{R}$ and σ_j^2 small enough we can locally linearize $\mathbf{g}'(\cdot)$ and obtain, from Proposition 6.1 and after some calculations, the approximated *pdf* of the map with the newly initialized landmark as the sum of Gaussians

$$p\left(\begin{bmatrix} X \\ \mathbf{p} \end{bmatrix} | \mathbf{y} \right) = \sum_{j=1}^{N_g} c_j' \cdot \mathcal{N}\left(\begin{bmatrix} X-\bar{X} \\ \mathbf{p}-\bar{\mathbf{p}}_j \end{bmatrix} ; \begin{bmatrix} \mathbf{P} & \mathbf{P}_{pX,j}^\top \\ \mathbf{P}_{pX,j} & \mathbf{P}_{pp,j} \end{bmatrix} \right) \tag{6.5}$$

where the elements for all hypotheses $1 \le j \le N_g$ are:

$$\begin{aligned}
c_j' &= c_j \; |[\mathbf{G}_{\mathbf{b},j} \; \mathbf{G}_{s,j}]|^{-1} && (6.6) \\
\bar{\mathbf{p}}_j &= \mathbf{g}(\bar{\mathcal{R}},\mathbf{y},s_j) && (6.7) \\
\mathbf{P}_{pX,j} &= \mathbf{G}_{\mathcal{R},j}\, \mathbf{P}_{\mathcal{R}X} && (6.8) \\
\mathbf{P}_{pp,j} &= \mathbf{G}_{\mathcal{R},j}\, \mathbf{P}_{\mathcal{R}\mathcal{R}}\, \mathbf{G}_{\mathcal{R},j}^\top + \mathbf{G}_{\mathbf{b},j}\, \mathbf{R}\, \mathbf{G}_{\mathbf{b},j}^\top + \mathbf{G}_{s,j}\, \sigma_j^2\, \mathbf{G}_{s,j}^\top && (6.9)
\end{aligned}$$

with $\bar{\mathcal{R}}$ the robot's pose conditional mean and where the Jacobian matrices are defined by

$$\mathbf{G}_{\mathcal{R},j} = \frac{\partial \mathbf{g}}{\partial \mathcal{R}^\top}\Big|_{\bar{\mathcal{R}},\mathbf{y},s_j} \qquad \mathbf{G}_{\mathbf{b},j} = \frac{\partial \mathbf{g}}{\partial \mathbf{b}^\top}\Big|_{\bar{\mathcal{R}},\mathbf{y},s_j} \qquad \mathbf{G}_{s,j} = \frac{\partial \mathbf{g}}{\partial s}\Big|_{\bar{\mathcal{R}},\mathbf{y},s_j}.$$

Doing

$$\bar{X}_j^+ = \begin{bmatrix} \bar{X} \\ \bar{\mathbf{p}}_j \end{bmatrix} \qquad \mathbf{P}_j^+ = \begin{bmatrix} \mathbf{P} & \mathbf{P}_{pX,j}^\top \\ \mathbf{P}_{pX,j} & \mathbf{P}_{pp,j} \end{bmatrix}$$

the map can now be rewritten in compact form for the sake of clarity

$$p(X^+|\mathbf{y}) = \sum_{j=1}^{N_g} c'_j \cdot \mathcal{N}(X^+ - \bar{X}^+_j; \mathbf{P}^+_j). \tag{6.10}$$

Now pay attention to expression (6.10): we clearly see the effect of our poor knowledge on s! To define the *pdf* of the new map, we need to define as many triplets $\{c'_j, \bar{X}^+_j, \mathbf{P}^+_j\}$ as there were terms in the initial sum of $p(s)$: we find that our map is now a set of N_g weighted Gaussian maps $X^+_j \sim \mathcal{N}\{\bar{X}^+_j; \mathbf{P}^+_j\}$ that define our whole knowledge about the system. Each Gaussian map is generated following the standard procedure for the totally observable case explained in (4.2.1), assuming that for each one of them the set $\{\mathbf{y}, s_j; \mathbf{R}, \sigma^2_j\}$ acts as a proper, Gaussian full observation, *ie.*

$$\begin{bmatrix}\mathbf{b}\\ s\end{bmatrix} \sim \mathcal{N}\left\{\begin{bmatrix}\mathbf{y}\\ s_j\end{bmatrix}; \begin{bmatrix}\mathbf{R} & 0\\ 0 & \sigma^2_j\end{bmatrix}\right\}.$$

In simpler words, trying to initialize a partially observed landmark has multiplied the complexity of our problem by N_g. In the case of simultaneous initialization of multiple (say m) landmarks, we would fall into the situation of having to maintain as much as $(N_g)^m$ maps. The map management would have to utilize the standard GSF, but such an exponential increase of the problem size makes this solution intractable in practice.

6.4 Federated Information Sharing SLAM

We need to find a computationally compelling alternative to GSF. Following the same multi-hypothesis reasoning, we can consider that each hypothesis corresponds to a different landmark. We can then initialize them all in one single Gaussian map using the standard EKF-SLAM procedure of Section 4.2.1. The result is a map that has grown in an additive way, avoiding the undesired multiplicative effect.

Divergence and inconsistency risks that emerge from the fact of having all hypotheses correlated in a unique map need to be minimized. For that, the hereafter proposed FIS technique firstly mimics the weight propagation scheme of the GSF, but then relies on these weights to adequately regulate the effect of the subsequent EKF corrections. The weights are also used to progressively eliminate the wrong hypotheses, allowing the ray to converge to a single Gaussian representation as the camera motion increases 3D observability.

This section first proposes a minimal representation for the Gaussian sum that defines the range's *pdf*. This will minimize the number of hypotheses. Then it goes on detailing the FIS initialization method, which could be seen as a shortcut of the more proper GSF-SLAM.

6.4.1 The ray: a geometric series of Gaussians

We look for a minimal implementation of (6.3), a safe way to fill the conic-shaped ray with the minimum number of Gaussian-shaped densities. For that, we start by giving the general realization of (6.2):

$$\mathbf{p} = \mathbf{x} + s \cdot \mathbf{R}(\mathbf{q}) \cdot \mathrm{dir}(\mathbf{b}) \tag{6.11}$$

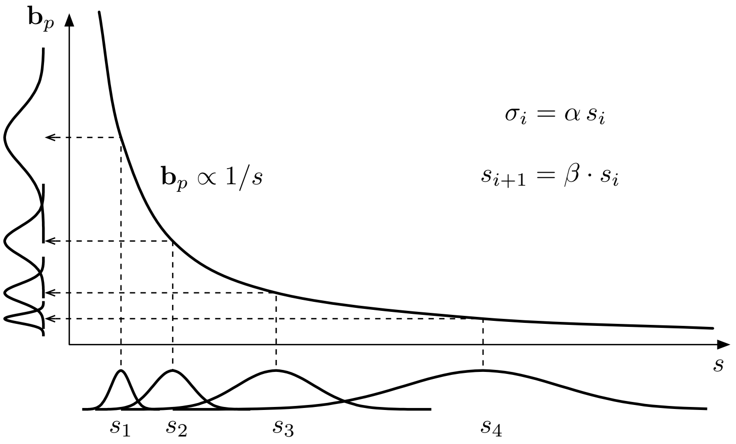
$\mathbf{b}_p$
$\sigma_i = \alpha\, s_i$
$\mathbf{b}_p \propto 1/s$
$s_{i+1} = \beta \cdot s_i$
s
s_1
s_2
s_3
s_4

Figure 6.4: Linearizations in the presence of the geometric series for a function $\mathbf{b} \propto 1/s$. When the standard deviation to mean ratio α is constant the linearization errors are too. We limit these errors by setting this ratio $\alpha < 0.3$.

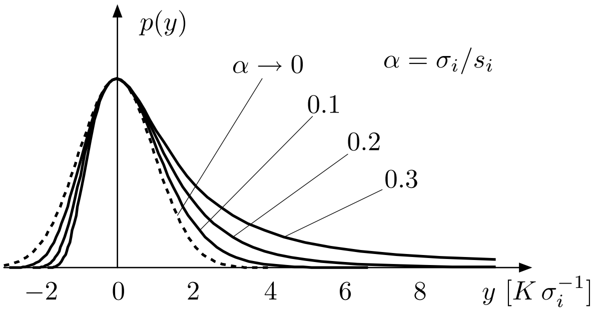

Figure 6.5: Normalized density asymmetry after transforming the Gaussian $\mathcal{N}(s - s_i, \sigma_i^2)$ through the function $y = K/s$ and centering. This asymmetry only depends on $\alpha = \sigma_i/s_i$ and is used to evaluate the linearization validity. The value $\alpha = 0.3$ shows a long tail but gave satisfying results in our experiments. It must be regarded as an extreme, absolute upper bound.

where dir($\mathbf{b}$) is a direction vector in sensor frame defined by $\mathbf{b}$;[6] $\mathbf{x}$ is the sensor position; $\mathbf{R}(\mathbf{q})$ is the rotation matrix associated with the sensor orientation $\mathbf{q}$; and s is the range, now unknown. We then remark that given a sensor in $\mathcal{S} = (\mathbf{x}, \mathbf{q})$ and a landmark in $\mathbf{p}$, the observed $\mathbf{b}$ is inversely proportional to s, as it can also be seen in the observation function $\mathbf{h}()$ for the case of vision:

$$s\,\underline{\mathbf{b}} = s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X^{\mathcal{S}} \\ Y^{\mathcal{S}} \\ Z^{\mathcal{S}} \end{bmatrix} = toFrame(\mathcal{S}, \mathbf{p}) \;\Rightarrow\; \mathbf{b} \propto 1/s \tag{6.12}$$

It is shown in [Kronham 1998; Peach 1995] that in such cases EKF is only relevant if the ratio $\alpha_j = \sigma_j/s_j$ is small enough (we use up to 30% in practice), as it determines the validity of the linearizations (Figs. 6.4 and 6.5). This leads to define $p(s)$ as a geometric series with

[6] Let $\mathbf{b} = [x, y]^\top$ be the metric coordinates of a pixel in a camera with focal length f. We have dir($\mathbf{b}$) $= [x/f, y/f, 1]^\top$. Landmark depth is denoted by s (Chapter 2).

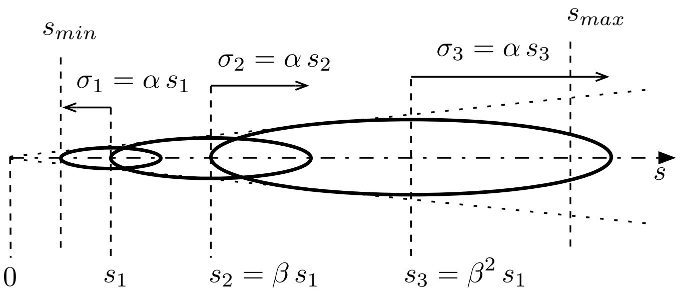
s_{min}
s_{max}
$\sigma_1 = \alpha\, s_1$
$\sigma_2 = \alpha\, s_2$
$\sigma_3 = \alpha\, s_3$
s
0
s_1
$s_2 = \beta\, s_1$
$s_3 = \beta^2\, s_1$

Figure 6.6: The conic Ray: a geometric series of Gaussian densities.

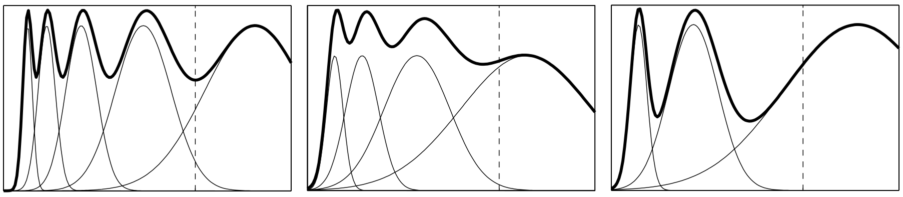

Figure 6.7: Geometric densities for $s_{min}/s_{max} = 10$: *Left:* $(\alpha, \beta) = (0.2, 1.8)$. *Center:* $(\alpha, \beta) = (0.3, 2)$. *Right:* $(\alpha, \beta) = (0.3, 3)$. The dotted line is at s_{max}.

$\alpha_j = \alpha =$ constant:

$$p(s) = \sum_{j=1}^{N_g} c_j \cdot \mathcal{N}(s - s_j, \sigma_j^2) \tag{6.13}$$

$$s_j = \beta \cdot s_{j-1} \tag{6.14}$$

$$\sigma_j = \alpha \cdot s_j. \tag{6.15}$$

An overview of the series with its parameters is shown in Fig. 6.6. From the bounds $[s_{min}, s_{max}]$, and the choice of the ratio α and the geometric base β, we need to determine the first term (s_1, σ_1) and the number of terms N_g. We impose the conditions $s_1 - \sigma_1 = s_{min}$ and $s_{N_g} + \sigma_{N_g} \geq s_{max}$ to get

$$s_1 = (1-\alpha)^{-1} \cdot s_{min} \tag{6.16}$$

$$\sigma_1 = \alpha \cdot s_1 \tag{6.17}$$

$$N_g = 1 + \text{ceil}\left[\log_\beta\left(\frac{1-\alpha}{1+\alpha} \cdot \frac{s_{max}}{s_{min}}\right)\right] \tag{6.18}$$

where ceil(x) is the next integer to x. We see that for this series to have a finite number of terms we need $s_{min} > 0$ and $s_{max} < \infty$, that is the limited depth range $s \in [s_{min}, s_{max}]$.

The geometric base β determines the sparseness of the series. Fig. 6.7 shows plots of the obtained *pdf* for different values of α and β. The couple $(\alpha, \beta) = (0.3, 3)$ defines a series that is somewhat far from the original uniform density, but experience showed that the overall performance is not degraded and the number of terms is minimized. In fact, and as it is shown in [Lemaire et al. 2005b], the system's sensibility to α and β is small and hence the choices are not that critical.

Table 6.1: Number of Gaussians for $\alpha = 0.3$ and $\beta = 3$.

Scenario	s_{min} (m)	s_{max} (m)	$\frac{s_{max}}{s_{min}}$	N_g
Indoor	0.5	5	10	3
Outdoor	1	100	100	5
Long range	1	1000	1000	7

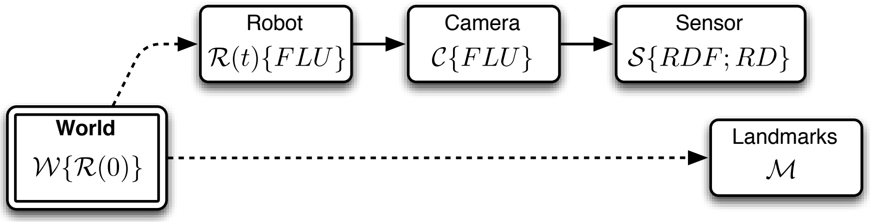

Figure 6.8: Reference frames diagram in FIS-SLAM (refer to Chapter 1 for notation). Solid-line arrows indicate deterministic definitions. Dashed-line ones are uncertain; their *pdfs* are continuously estimated by FIS-SLAM.

Table 6.1 shows the number of Gaussians for three typical applications. Note how, thanks to the geometric series, increasing s_{max}/s_{min} by a factor 10 implies the addition of just two members.

6.4.2 The FIS algorithm

We use the world representation sketched in Fig. 6.8. We have to distinguish two types of landmarks in our map of this world:

- Landmarks which are represented by a single Gaussian
- Landmarks which are represented by a ray, the set of Gaussians.

As a SLAM method, FIS-SLAM has the objective to keep the map up to date when the following situations occur (Fig. 6.9):

1. The robot observes a landmark already existing in the map in the form of a single Gaussian;
2. The robot observes a landmark already existing in the map in the form of a ray;
3. The robot observes a new landmark and decides to incorporate it in the map; and
4. The robot moves.

Robot motion and the observations of purely Gaussian landmarks follow the classical EKF-SLAM formulation and are not explained here. The particularity of FIS-SLAM resides in the way to initialize the ray describing the landmark *pdf* and the way to perform updates on these rays, while making them converge to pure Gaussians as full observability is getting achieved.

The aim of the rays management in FIS-SLAM is therefore twofold: we want to safely and progressively select the Gaussian in the ray that best represents the real landmark, while

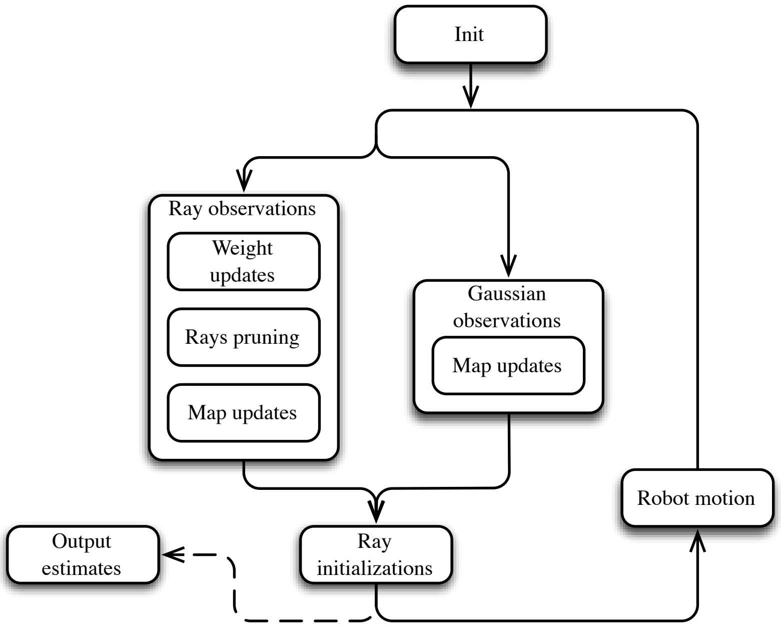

Figure 6.9: The FIS-SLAM algorithm.

using at the same time the angular information this ray provides. It consists of three main operations: *1)* the inclusion of all the members of the ray into the map; *2)* the successive pruning of bad members;[7] and *3)* the map updates using Federated Information Sharing. Fig. 6.10 gives a compact view of the whole process.

FIS: Iterated ray initialization

As discussed earlier, we include all landmark hypotheses that conform the ray in a single Gaussian map. All ray members $\{\mathbf{p}_1 \cdots \mathbf{p}_{N_g}\}$ are stacked in the same random state vector as if they corresponded to different landmarks:

$$X^+ = \begin{bmatrix} X \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_{N_g} \end{bmatrix}. \tag{6.19}$$

An iterated method is used to construct its mean and covariances matrix (Fig. 6.11). Landmark hypotheses are stacked one by one by iteratively applying the initialization procedure of EKF-SLAM (Section 4.2.1), by considering the 'full measure' $\{\mathbf{y}, s_j; \mathbf{R}, \sigma_j^2\}$ indicated before:

$$\begin{bmatrix} \mathbf{b} \\ s \end{bmatrix} \sim \mathcal{N}\left\{\begin{bmatrix} \mathbf{y} \\ s_j \end{bmatrix}; \begin{bmatrix} \mathbf{R} & 0 \\ 0 & \sigma_j^2 \end{bmatrix}\right\}.$$

[7] This operation requires a sub-division into smaller steps as it will be seen.

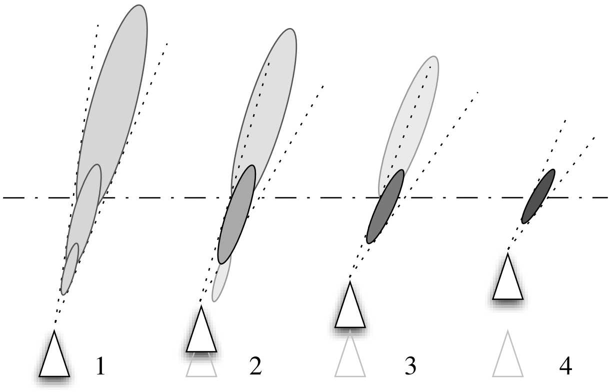

Figure 6.10: Ray updates on 4 consecutive poses. Gray level indicates the weights that are used to discard bad hypotheses and to weight the corrections on surviving ones. The dash and dot line is at the true distance to the landmark.

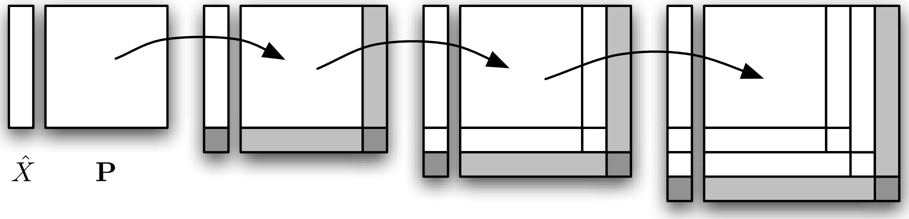

Figure 6.11: Iterated Ray Initialization for $N_g = 3$. Each arrow states for an EKF-SLAM-based landmark initialization.

The result looks like this:

$$\bar{X}^+ = \begin{bmatrix} \bar{X} \\ \bar{\mathbf{p}}_1 \\ \vdots \\ \bar{\mathbf{p}}_{N_g} \end{bmatrix} \qquad \mathbf{P}^+ = \begin{bmatrix} \mathbf{P} & \mathbf{P}_{pX,1}^\top & \cdots & \mathbf{P}_{pX,N_g}^\top \\ \mathbf{P}_{pX,1} & \mathbf{P}_{pp,1} & & \\ \vdots & & \ddots & \\ \mathbf{P}_{pX,N_g} & & & \mathbf{P}_{pp,N_g} \end{bmatrix}. \tag{6.20}$$

Initially, all hypotheses are given the same credibility so their weighting must be uniform. We write the uniform weights vector associated with the newly added ray:

$$\Lambda = \begin{bmatrix} \Lambda_1^* & \cdots & \Lambda_{N_g}^* \end{bmatrix} \quad ; \quad \Lambda_j^* = 1/N_g. \tag{6.21}$$

The evolution of these weights will reflect the likelihood of each hypothesis with the whole historic of measurements. This we see in the next paragraph.

FIS: Ray weights updates

As in the standard GSF, the weight Λ_j of each hypothesis is successively updated with its measure of likelihood λ_j. In Chapter 3 we defined this likelihood as the probability of hypothesis j being true given the observation $\{\mathbf{y}; \mathbf{R}\}$. When the observation is two-dimensional, this likelihood becomes:

$$\lambda_j \triangleq \mathcal{N}(\mathbf{z}_j; \mathbf{Z}_j) = \frac{1}{2\pi\sqrt{|\mathbf{Z}_j|}} \exp(-\frac{1}{2}\mathbf{z}_j^\top \mathbf{Z}_j^{-1} \mathbf{z}_j)$$

with $\{\mathbf{z}_j; \mathbf{Z}_j\}$ the innovation and its covariances matrix:

$$\begin{aligned} \mathbf{z}_j &= \mathbf{y} - \mathbf{h}(\bar{\mathcal{R}}, \bar{\mathbf{p}}_j) \\ \mathbf{Z}_j &= \mathbf{H}_j\, \mathbf{P}_j\, \mathbf{H}_j^\top + \mathbf{R} \end{aligned}$$

and $\mathbf{H}_j$ the Jacobian matrix of the observation function:

$$\mathbf{H}_j = \left.\frac{\partial \mathbf{h}}{\partial X^\top}\right|_{\bar{\mathcal{R}}, \bar{\mathbf{p}}_j}.$$

We define the weight of a hypothesis as the probability of being true given its *a priori* probability Λ_j^* and all observations up to time k:

$$\Lambda_j(k) \triangleq \Lambda_j^* \prod_{t=0}^{k} \lambda_j(t) \tag{6.22}$$

where the *a priori* weight $\Lambda_j^* = 1/N_g$ is known. This can be written in recursive form as

$$\Lambda_j^+ = \Lambda_j \cdot \lambda_j. \tag{6.23}$$

The weights are systematically normalized so that $\sum_j \Lambda_j = 1$.

FIS: Weights equalization

The last two equations define a weight of *uniform horizon*, *i.e.* the whole history of likelihood uniformly contribute to the hypothesis weight. Experimental evidence showed that this uniform horizon is not very convenient to our purposes, as we examine now. We said (Chapter 5) that the initialization process for landmarks that are either *–very distant–* or *–close to the motion axis–* can potentially be very long. In these cases one has to additionally consider and correct the effect of *weights dispersion*, an undesired risk of drift in the values of the weights distribution when the multi-hypothesized dimension is not observable for a relatively long time. In effect, consider the weight definition (6.22) and rewrite it as

$$\Lambda_j(k) = \Lambda_j^* \cdot (2\pi)^{-\frac{(k+1)n}{2}} \cdot \left(\prod_{t=0}^{k} |\mathbf{Z}_j|\right)^{-\frac{1}{2}} \cdot \exp\left(-\frac{1}{2}\sum_{t=0}^{k}(\mathbf{z}_j^\top \mathbf{Z}_j \mathbf{z}_j)\right)$$

and concentrate on the exponential part (which is predominant). Observe that the exponent

$$V_j \triangleq \sum_{t=0}^{k}(\mathbf{z}_j^\top \mathbf{Z}_j \mathbf{z}_j)$$

is a sum of a quadratic form of the innovation $\{\mathbf{z}_j; \mathbf{Z}_j\}$, which is close to a white Gaussian noise $\mathcal{N}(\mathbf{z}_j; \mathbf{Z}_j)$. One can argue that when k tends to infinity this sum will have a standard deviation infinitely larger than its mean (just make the simile with the Brownian motion, which is the sum of white Gaussian noise: its variance tends to infinity while its means remains zero). If all innovation pairs were independent, this would lead to a dispersion of the V_j which would in turn induce a dispersion of the weights Λ_j. On the contrary, if all $\{\mathbf{z}_j; \mathbf{Z}_j\}$ were exactly equal, all V_j would evolve exactly in the same manner, leading to an absolute stability of the weights distribution. In the general case the $\{\mathbf{z}_j; \mathbf{Z}_j\}$ are not independent but there is nothing that guarantees that they are equal, so the null dispersion cannot be guaranteed. Dispersion then means that there is a real chance that one weight takes a value arbitrarily larger than the other ones. When normalized, this weight will tend to the unit value and the other ones will consequently vanish.

Weights dispersion is in fact an advantage for observable systems as the filter will provably tend to a single-member one (*ie.* one EKF!), the choice of the right member being driven by the observations. But it is catastrophic for poorly observable systems, specially if one wishes to prune the weakest members as we do. If the landmark's depth is not observable for a certain amount of time we can easily prune the right member, something that can not be undone. Thus this dispersion must be definitely avoided. This can be very easily done by vanishing the effect of very old observations on the weights so as to have a *vanishing horizon*

$$\Lambda_j(k) \triangleq \Lambda_j^* \prod_{t=0}^{k} \big(\lambda_j(t)\big)^{(1-\gamma)(k-t)} \tag{6.24}$$

instead of the uniform (6.22). In recursive form, this equalizing weights update becomes:[8]

$$\Lambda_j^+ = \Lambda_j^{(1-\gamma)}\,\lambda_j, \tag{6.25}$$

where $0 \leq \gamma \leq 1$ is called the *vanishing factor*. Its effect is a slight tendency to bring all weights back to the uniform $1/N$ value, therefore the term *weights equalization*. The value of γ must be chosen so that this tendency is surpassed by the observations effect on the likelihood under the conditions of sufficient observability.

A more proper analysis of this phenomenon has not been done (we believe that the subject merits a deeper insight and that it would generate a discourse which is not suited for this thesis[9]) and the values of γ will have to be set heuristically. We provide in Section 6.4.3 some simulations under conditions of poor observability that will help to clarify things up.[10]

FIS: Ray members pruning

The *divergence* risk calls for a conservative criterion for pruning those members with very low weight. This will in turn allow the ray to collapse to a single Gaussian.

We consider a ray of N members and use the weights vector $\Lambda = [\Lambda_1, \cdots, \Lambda_N]$ which is normalized so that $\sum_j \Lambda_j = 1$. For pruning, we compare the member's weights against the uniform weight $1/N$ by means of a simple threshold τ. Ray member j is deleted if the ratio $\frac{\Lambda_j}{1/N} < \tau$, or equivalently if

$$N \cdot \Lambda_j < \tau \tag{6.26}$$

where τ is in the range $\begin{bmatrix}0.0001 & 0.01\end{bmatrix}$, typically 0.001, and roughly corresponds to the probability of pruning a valid hypothesis (pretty similar to Sequential Probability Ratio Test in [Kwok and Dissanayake 2004] for example). At each member deletion we update $N^+ = N - 1$. When $N = 1$, we say the ray has *collapsed* to a single Gaussian and so it will be treated as in standard EKF-SLAM.

[8] Notice that this equation performs both weight update and equalization and hence it substitutes (6.23).

[9] For example, for a linear-Gaussian system, the evolution of the weights is absolutely non-linear. When these weights are considered as part of the system variables, the linear system becomes non-linear.

[10] Weights dispersion is conceptually very close to the phenomenon of particle degeneracy in the PF (Chapter 3), but the mechanisms and the conditions under which they appear are not the same, notably because of the fact that the particle innovations can effectively be considered independent while in this case they are strongly dependent.

FIS: Ray twin members merging or pruning

Ray pruning is an effective procedure to eliminate those hypothesis that are more or less far from representing the real landmark. We look now at the two hypothesis which are closest to the true landmark: one is between the robot and the landmark and the other is beyond the landmark. As they are updated at each observation (see the next paragraph: Map updates), they move along the ray axis and they approach the landmark position. Very often they survive the pruning process and they end up confounded at the same point, hence the term *twin members*. This is detected when both depths differ in less than a certain percentage (for example 10%).

At this time we can apply merging: the parameters of both Gaussians (weight, mean and covariances) are combined together to result in a single Gaussian. Alternatively, we can opt to simply prune the less likely Gaussian, an operation which is much easier to implement and perfectly safe, because the variances in the ray axes are still large compared to their depths difference.

FIS: Map updates via Federated Information Sharing

This is the most delicate stage. We have a fully correlated map with all hypotheses in it, so a correction step on one hypothesis has an effect over the whole map. If the hypothesis is wrong, this effect may cause the map to *diverge*.

Of course we would like to use the observation to correct the map at the right hypothesis. As we don't know which one it is, we are obliged to actuate on all of them. This involves the risk of *inconsistency*: if we incorporate multiple times the same information (remember that we have a unique observation for all hypotheses), the map covariance $\mathbf{P}$ will shrink according to the multiple application of the EKF correction equations (4.13), leading to an overconfident estimate of the map X.

The proposed FIS method is inspired by the Federated Filter (FF) in [Foxlin 2002] to address these problems. FF is a decentralized Kalman filter that allows a paralleled processing of the information. In the case this information comes from a unique source, as it is our case, FF applies the Principle of Measurement Reproduction [Tupysev 1998] to overcome *inconsistency*. This principle can be resumed as follows: The correction of the estimate of a random variable by a set of measurement tuples $\{\mathbf{y}; \mathbf{R}_j\}$ is equivalent to the unique correction by $\{\mathbf{y}; \mathbf{R}\}$ if

$$\mathbf{R}^{-1} = \sum_j \mathbf{R}_j^{-1}, \tag{6.27}$$

which simply states that the sum of information for all corrections must equal the information provided by the observation. This is what is done by FIS. The idea (Fig. 6.12) is to share the information given by the observation tuple $\{\mathbf{y}; \mathbf{R}\}$ among all hypotheses. Doing $\mathbf{R}_j = \mathbf{R}/\rho_j$, condition (6.27) is satisfied if

$$\sum_j \rho_j = 1. \tag{6.28}$$

The scalars ρ_j get the name of *federative coefficients*.

The *divergence* risk is also addressed by FIS. We need to choose a particular profile for ρ_j that privileges the corrections on more likely hypotheses. A flexible way to do so is to make it monotonically dependent on each hypothesis weight Λ_j by taking $\rho_j \propto (\Lambda_j)^\mu$, where μ is a

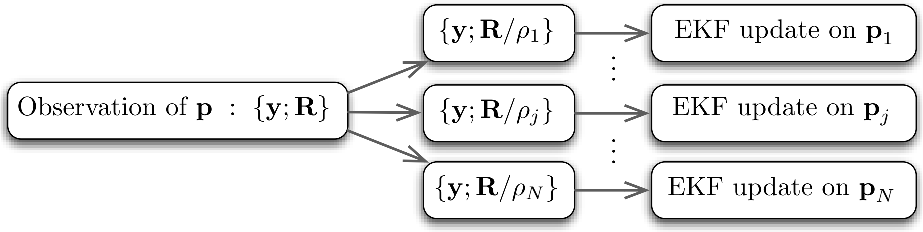
Observation of $\mathbf{p}$: $\{\mathbf{y};\mathbf{R}\}$
$\{\mathbf{y};\mathbf{R}/\rho_1\}$
$\{\mathbf{y};\mathbf{R}/\rho_j\}$
$\{\mathbf{y};\mathbf{R}/\rho_N\}$
EKF update on $\mathbf{p}_1$
EKF update on $\mathbf{p}_j$
EKF update on $\mathbf{p}_N$

Figure 6.12: Map update via Federated Information Sharing.

shape parameter that permits us to regulate how much we want to privilege strong hypotheses over weak ones. These two conditions on ρ_j lead to

$$\rho_j = \frac{(\Lambda_j)^\mu}{\sum_{j=1}^N (\Lambda_j)^\mu}. \tag{6.29}$$

The shape parameter μ takes appropriate values between $\mu = 1$ and $\mu = 3$. In the typical case $\mu = 1$ the federative coefficient for member j is exactly its normalized weight $\rho_j = \Lambda_j$. The value $\mu = 0$ uniformly distributes the importance of all corrections. On the contrary, correcting only on the biggest weight hypothesis as it is done by Kwok and Dissanayake [2004] means taking $\mu \to \infty$.[11]

FIS: Output estimates

Should an optimal estimate be needed, the minimum of variance estimate is obtained as follows. For robot pose and Gaussian landmarks we simply have

$$\hat{\mathcal{R}} = \bar{\mathcal{R}} \tag{6.30}$$

$$\hat{\mathbf{p}}_i = \bar{\mathbf{p}}_i \tag{6.31}$$

For a ray i with members j we can do

$$\hat{\mathbf{p}}_i = \sum_j \Lambda_{i,j}\, \bar{\mathbf{p}}_{i,j}, \tag{6.32}$$

but it may be worth noticing that a punctual estimate of a ray might not be of any use, or at least that it should be interpreted with care.

6.4.3 Simulations

Poor observability simulations

This simulation illustrates the evolution of the main system variables during lack of observability. For this, a simplified system is set up as follows (Fig. 6.13). In 2D, a bearings-only sensor is placed at the origin of coordinates, heading towards the X axis. In this same axis a landmark is situated at 5m of distance. Two hypotheses are generated for the landmark

[11] There is a slight difference: in [Kwok and Dissanayake 2004] the update is made taking the most likely hypothesis only with respect to the last observation. This would mean using λ_j instead of Λ_j in (6.29) with $\mu \to \infty$.

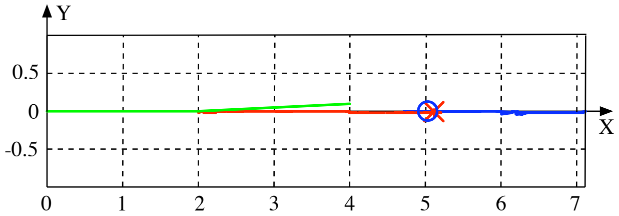

Figure 6.13: Poor observability simulation set-up. The camera trajectory is plotted in *green*. The landmark is at $(5, 0)$. The first hypothesis starts at $(2, 0)$ and moves along the X axis, producing the *red* track, until it ends up at the red cross. The second one, in *blue*, starts at $(6, 0)$ and ends up at the blue circle. The figure is difficult to appreciate precisely because of the poor observability.

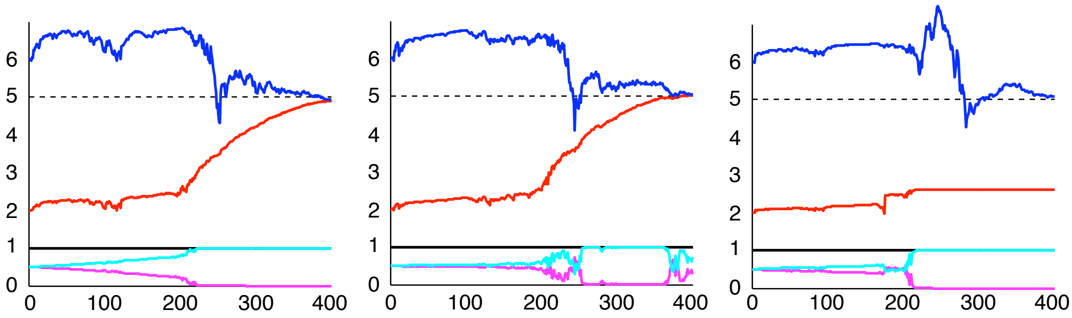

Figure 6.14: Weights dispersion and its correction. X-coordinates of the landmark's first (*red*) and second (*blue*) estimates. Respective normalized weights are in *cyan* and *magenta*. Up to time 200 the trajectory is exactly singular and the landmark depth not observable. From then on the small turn of 2° to the left provides a weak observability that is adequately exploited by the filters. *Left:* The two EKF's non-equalized weights and estimates. *Center:* The operation of *weights equalization* is added. *Right:* With weights equalization and federated map updates as in the FIS algorithm.

position: one at 2m and the other one at 6m using the geometric pair $(\alpha, \beta) = (0.3, 3)$. The sensor starts moving towards the landmark (precisely on the X axis), and after 2m it slightly turns an angle of 2° to the left; then it continues for two more meters. Two weighted EKF are set up, one for each landmark hypothesis. The camera motion is considered deterministic; angular measurement errors are 0.6° standard deviation.

We are interested in seeing the behavior of our variables (filter states and weights) during the unobservable period and during the transition to observability (Fig 6.14).

As we already know, during the first 2m the landmark distance is unobservable. Both estimates move erratically and do not show a particular behavior. On the contrary, the weights accumulate this erratic phenomenon and tend to diverge (Fig. 6.14 *left*). If the system is run in this way for a longer time, one of the weights will fall below the pruning threshold and the hypothesis it represents will be deleted. This, when the system is unobservable, is a bad thing to do.[12]

We correct it by adding the weights equalization step given by (6.25) with $\gamma = 0.05$ (Fig. 6.14 *center*). The divergent trend is compensated and both hypotheses survive until observability allows for consequent decisions.

Finally the case of federated updates is considered in Fig. 6.14 *right.* The hypothesis with

[12] You should not make decisions without having trustable information.

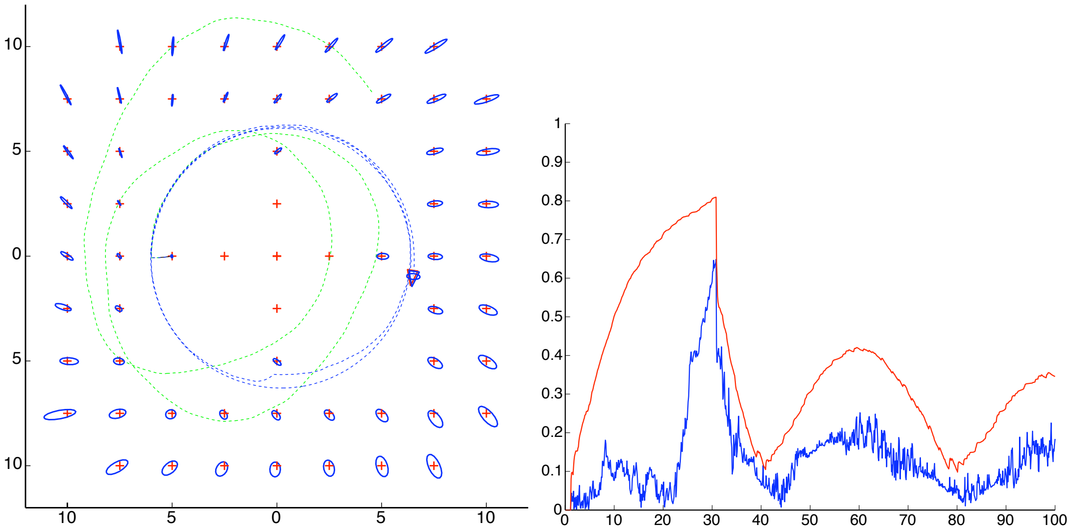

Figure 6.15: Indoor simulation. *Left*: Circular trajectory. *Right*: Consistency test: 3σ bound robot position error estimates (red) vs. true error (blue).

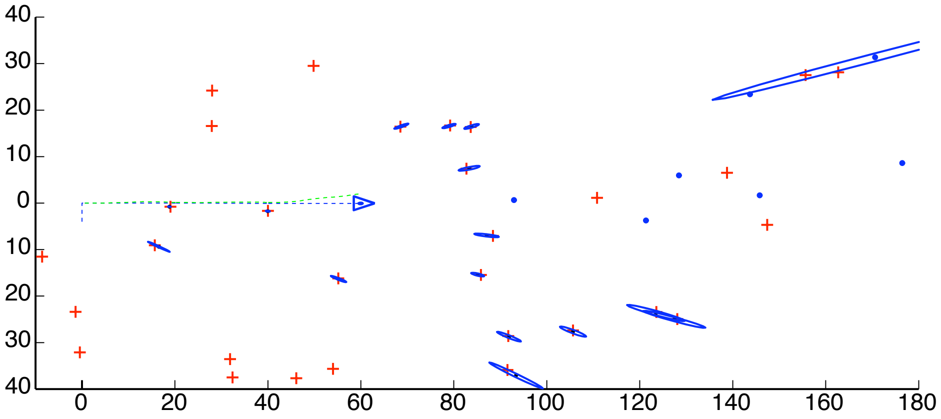

Figure 6.16: Outdoor area, straight trajectory simulation.

lower weight is less corrected and does not converge to the true landmark position, something that will help to discriminate it. The surviving one converges correctly.

Full algorithm simulations

Simulations with a two-dimensional implementation have been carried out to validate the proposed methods in more complex, realistic situations. In the following figures we illustrate the results of these simulations. Ground truth landmarks are represented by red crosses. Small or elongated blue ellipses represent the 3σ-bound regions of the landmark Gaussian estimates. Estimated trajectories are plotted in dashed blue line. Odometry integration trajectories are plotted in dotted green line. All simulations use $\alpha = 0.3$, $\beta = 3$, $\gamma = 0$, $\mu = 1$ and $\tau = 0.001$. They can be appreciated as video sequences in `http://www.laas.fr/~jsola/objects/videos/PhD/video-6N.mov` , where `N` is a video number, or in the included CD.

Results from simulations on two different scenarios are given. In the first (Fig. 6.15 *left*), a robot makes two turns following a circular trajectory inside a square cloister of some 20m in size, where the columns are treated as landmarks. Linear and angular speeds are 1m/s and 0.16rad/s. Odometry errors are simulated by corrupting these values with white Gaussian noises with standard deviations of 0.3m/s and 0.3rad/s respectively. Bearings are acquired every 100ms with a sensor that is looking forward with a field of view of ±45° and an accuracy of 1°. A consistency test that plots the 3σ-bound estimated error for the robot position against the true error is given in Fig. 6.15 *right*.

The second scenario (Fig. 6.16) simulates an outdoor area of 180x80m, populated with 30 randomly distributed landmarks. The robot follows a straight trajectory at a speed of 2m/s. Odometry errors are 0.1m/s and 0.1rad/s. Bearings are acquired every 100ms with a sensor that looks forward with a field of view of ±30° and an accuracy of 0.5°. Observe how, for landmarks close to the motion direction, the several hypotheses that are initialized (shown as isolated dots at their means, without the associated covariance ellipses) have not yet collapsed to single Gaussians.

6.5 Feature detection and matching using vision

In order to feed the FIS-SLAM algorithm, the luminance information provided by the cameras (the images) must be translated into geometric information in the 2D image plane (*i.e.* to know '*where*' the visual features are in the image). We take a sparse, punctual representation of the world in the form of landmarks, that are selected in the images in function of their interest or uniqueness (Chapter 2). The image processing algorithms responsible for these tasks (feature detectors and matchers) are usually computationally expensive. We take for this reason a 'top-down' approach known as 'active feature search' to target both feature localization in the images and their expected appearance, operations that are guided by the information we have been able to collect into the SLAM map so far (Chapter 5 and [Davison 2003], also referred to as *top-down* by Eade and Drummond [2006]). This allows for very robust operation with an enormous economy of resources: *a)* the feature detector only has to analyze the image regions where we know there are no features yet; and *b)* the feature matcher knows where and under which appearance the feature is, and the search is therefore performed in 2D space, inside a very tiny region, and with simple correlation-based methods. We explain now in detail all these perception procedures. Further justification can be found in Chapters 2 and 5.

6.5.1 Landmarks model

Each perceived landmark is modelled by both geometrical and photometrical information and by the initial viewpoint conditions.

The geometrical part consists in its Euclidean position in space, $\mathbf{p} = [p_x \; p_y \; p_z]^\top$, which is stochastically included in the SLAM map with the mentioned FIS initialization method.

The photometrical part is an appearance-based descriptor consisting of a medium-sized patch of about 15×15 to 21×21 pixels around the image of the point. This model, used in many real time vision-based SLAM works, is suitable for posterior feature matching by a maximum of correlation scan.

The initial viewpoint is just a copy of the robot pose (position and orientation) at the time of detection.

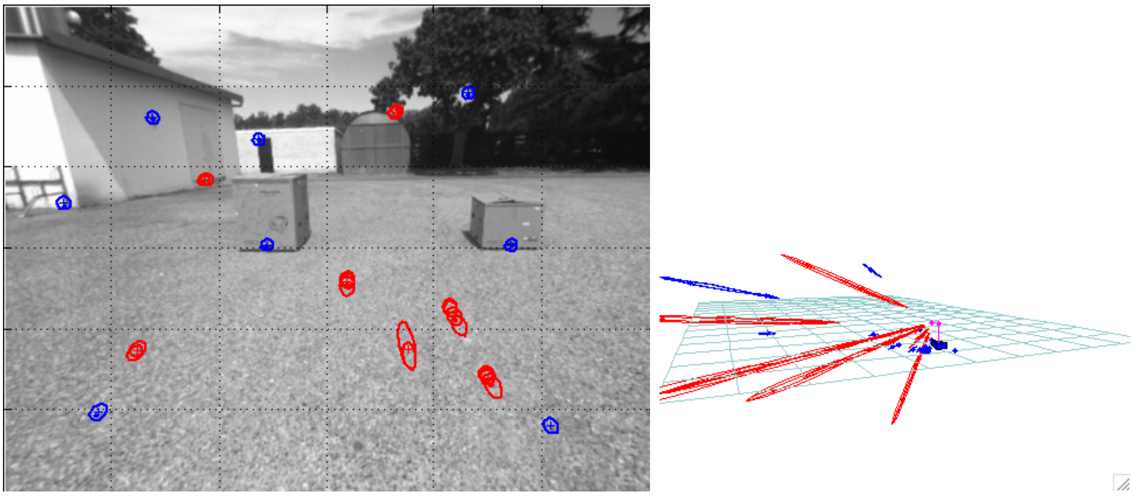

Figure 6.17: Image regions for new features search (*dotted grid*). Expectation ellipses for points (*blue*). Expectation overlapping ellipses for rays (*red*). Successful measures are plotted as tiny dots inside the expected ellipses (sorry, they are quite invisible!). The corresponding 3D map is shown.

6.5.2 Feature Detection

For feature detection, a heuristic strategy is used to select a region of interest in the image. We use (Fig. 6.17) a fixed grid that divides the image in a number of equal regions, and randomly select one (or more than one) grid element within those with no landmarks in it. The strongest Harris point [Harris and Stephens 1988] in this grid region is selected for landmark initialization. This naturally guarantees a uniform spreading of features within the image, where their density can be adjusted with the grid size. If the feature detection is satisfactory, its associated ray is calculated and initialized in the map. A medium-sized rectangular region or *patch* around the point is stored as the landmark's appearance descriptor, and the current pose of the robot is memorized.

6.5.3 Feature Matching (FM)

For feature matching (FM), we follow the *active search* approach, for which we reproduce the schematic sketch in Fig. 6.18, already presented in Chapter 5. This approach combines the simplicity of patch descriptors and correlation-based scans with the robustness of invariant matching: instead of invariant descriptors like [Lowe 2004; Shi and Tomasi 1994], we appropriately *vary* them before each scan using the information available in the map. False matchings are also drastically minimized as they will normally fall outside the predicted search regions. Active search is completed with an information-gain-driven strategy to select those landmarks that are most valuable to be measured.

Feature matching is accomplished in five differentiated steps:

1. Projection of the map information;

2. Information-gain selection;

3. Patch warping;

4. Correlation-based scan; and

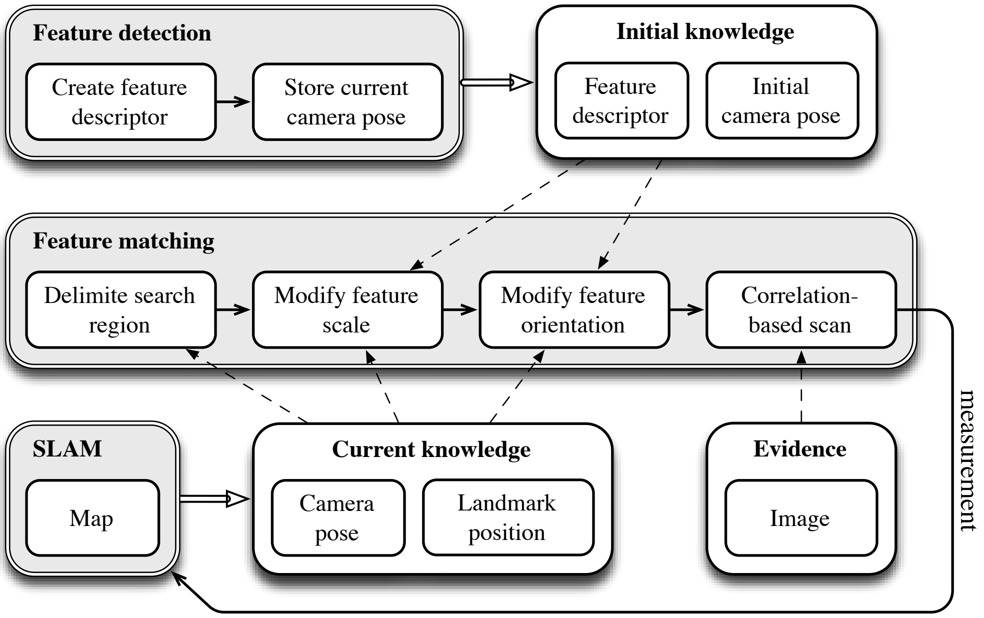

Figure 6.18: Active feature search (in the central gray thread) is guided by the available information (inside white boxes) which comes from the SLAM map and the feature detector, and by the current evidence provided by the perceived images.

5. Measurement validation.

Detailed descriptions are given below.

FM: Map projection: the expectations

The joint estimates of landmark and robot positions are projected into the image together with their covariances, obtaining what we call the *expectations* $\mathbf{e} \sim \mathcal{N}\{\bar{\mathbf{e}}; \mathbf{E}\}$ (revisit Fig. 6.17). As we deal with Gaussian *pdfs* this operation is performed for landmark i simply with

$$\begin{aligned} \bar{\mathbf{e}}_i &= \mathbf{h}(\bar{\mathcal{R}}, \bar{\mathbf{p}}_i) && (6.33) \\ \mathbf{E}_i &= \mathbf{H}_i \, \mathbf{P} \, \mathbf{H}_i^\top + \mathbf{R} && (6.34) \end{aligned}$$

or for each Gaussian j of a ray i with

$$\begin{aligned} \bar{\mathbf{e}}_{i,j} &= \mathbf{h}(\bar{\mathcal{R}}, \bar{\mathbf{p}}_{i,j}) && (6.35) \\ \mathbf{E}_{i,j} &= \mathbf{H}_j \, \mathbf{P} \, \mathbf{H}_j^\top + \mathbf{R} && (6.36) \end{aligned}$$

in any case a global estimate is computed with

$$\hat{\mathbf{e}}_i = \mathbf{h}(\hat{\mathcal{R}}, \hat{\mathbf{p}}_i). \qquad (6.37)$$

We want to notice here that these expectations are very closely related to the innovations: we have, in effect, for the case of a single Gaussian

$$\begin{aligned} \mathbf{z}_i &\triangleq \mathbf{y} - \mathbf{h}(\bar{\mathcal{R}}, \bar{\mathbf{p}}_i) \ = \mathbf{y} - \bar{\mathbf{e}}_i \\ \mathbf{Z}_i &\triangleq \mathbf{H}_i \, \mathbf{P} \, \mathbf{H}_i^\top + \mathbf{R} = \mathbf{E}_i. \end{aligned}$$

The expectations' mean $\bar{\mathbf{e}}_i$ and covariances matrix $\mathbf{E}_i$ are used to draw the 3σ ellipses in the image (or sets of ellipses in the case of rays) inside which the features will be searched.

FM: Information-gain-based feature selection

An information-gain approach is used to select the most interesting landmarks in the image to be measured. For this, the following criterion is used:

Criterion 6.1 (Information gain):

> *The information gain upon a map update following a landmark measurement is bigger for those landmarks with more uncertain expectation.* ◇

In other words, if the expectation is very precise (that is, we know with high precision where the landmark must be projected in the image), measuring that landmark will give little information.[13] If it is very uncertain, it will be worth measuring it.[14] Mathematically, this can be seen by exploring the evolution of the Kalman gain and the filter covariances matrix when the expectation uncertainty rises. Consider the measurement noise $\mathbf{R}$ to be constant: the expectation's uncertainty $\mathbf{E}_i$ is then directly related to $\mathbf{H}_i\mathbf{P}\mathbf{H}_i^\top$, the projection of the joint robot-landmark covariances matrix onto the image plane. The Kalman gain is defined

$$\mathbf{K}_i = \mathbf{P}\mathbf{H}_i^\top(\mathbf{H}_i\mathbf{P}\mathbf{H}_i^\top + \mathbf{R})^{-1},$$

and the filter covariances matrix evolves as

$$\mathbf{P}^+ = \mathbf{P} - \mathbf{K}\mathbf{H}_i\mathbf{P}$$

where the negative term is the gain in precision after an observation, thus directly related to the information gain. This term is

$$\Delta\mathbf{P} = \mathbf{P}\mathbf{H}_i^\top(\mathbf{H}_i\mathbf{P}\mathbf{H}_i^\top + \mathbf{R})^{-1}\mathbf{H}_i\mathbf{P}$$

which, being $\mathbf{H}_i$ bounded (fixed by the observation equation), is bigger when $\mathbf{H}_i\mathbf{P}\mathbf{H}_i^\top$ increases.

The expectation uncertainty, that we will denote ε_i, is evaluated by means of the determinant of its covariances matrix (Appendix B):

$$\varepsilon_i = |\mathbf{E}_i| \tag{6.38}$$

In the case of rays we just take, as a coarse approximation, the biggest determinant of all its members:

$$\varepsilon_i = \max_j |\mathbf{E}_{i,j}| \tag{6.39}$$

Two lists are generated (one of points and one of rays) with descending order of expectation's uncertainty. From these lists, the first predefined N_p points and N_r rays are selected to be measured.

[13] You don't need to check every five minutes if your keys are in your pocket: you know they are there...

[14] ...but you may need to check if your cat is at home before closing the door.

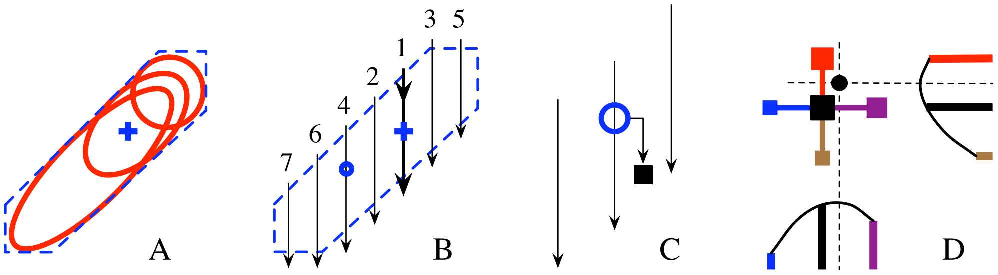

Figure 6.19: Correlation-based scan with sub-pixellic resolution. *A:* A region containing the 3σ ellipses of the landmark's expectations is defined. *B:* A global scan at double-pixel spacing is performed inside this region, starting at the expectation $\hat{\mathbf{e}}_i$ (*blue cross*) and spreading out symmetrically as arrow indices indicate, and the pixel with maximum ZNCC score is retained (*blue circle*). *C:* A local scan for a ZNCC maximum (black square) is performed at one pixel spacing around this pixel. *D:* Two parabolic interpolations, one in the horizontal- and one in vertical direction, with the ZNCC scores of the four cardinal neighbors of the winning pixel (*colored squares*) give an accurate sub-pixellic result (*black dot*).

FM: Linear patch warping

To perform the measurements the stored patch of each landmark is linearly warped (zoomed and rotated) the amounts defined by the change in the robot position as described in Chapter 5. The warped patch will have the most similar appearance as possible, given the knowledge we have on the system, to that in the current image. This will maximize the chances of a successful correlation-based match.

FM: Correlation-based scan with sub-pixellic resolution

A 2D search for the best correlation score of this modified patch inside the expectation's region gives the pixel that we take as the landmark's measurement. This comprises three sub-steps (Fig. 6.19): *a)* double-space global scan, *b)* single-space local scan, and *c)* sub-pixellic interpolation. Because of its simplicity and stability, the Zero-mean Normalized Cross-Correlation (ZNCC) is used as the appearance-based similarity measure.

A first global scan is performed inside the 3σ region defined by the ellipse (or the set of ellipses for rays). As we saw in Section 2.4.2 of Chapter 2, the peak of the ZNCC for good-defined features is always some pixels wide. This means that this scan can be done at double-pixel spacing and therefore that only 25% of the pixels in the region need to be evaluated. The pixel giving the highest ZNCC correlation score is retained.

A second local scan is performed at one pixel spacing at the vicinity of the pixel resulting from the global scan. This scan is guided by an increase of the ZNCC score.

The final measure is defined as the sub-pixellic coordinates of the maximum of the parabolic interpolations (vertically and horizontally) of the scores of this winning pixel with its four cardinal neighbors.

FM: Measurement validation

On return from the patch scanning two parameters are evaluated in order to validate the measurement.

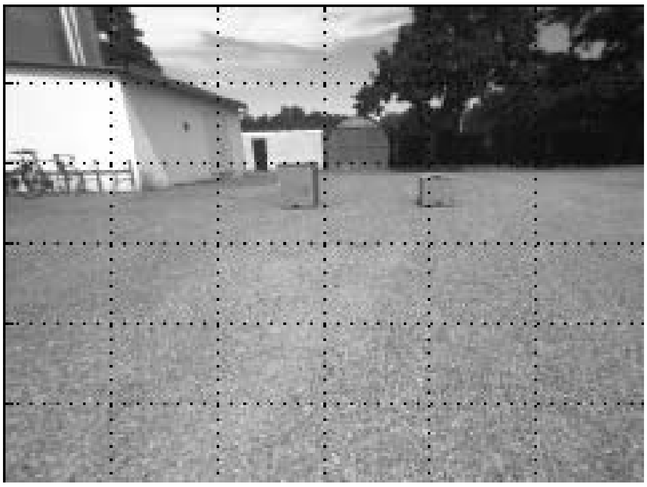

Figure 6.20: The ‘Boxes’ experiment. The robot will pass between the two boxes following a straight trajectory. The grid is used to guide the feature detector by searching new features only in those inner 4×4 grid cells without any feature.

First, the correlation score must be above a certain threshold to indicate that the appearance has not substantially changed and hence that the measured feature corresponds to the stored landmark descriptor. This test will fail usually when landmarks are occluded or when the camera viewpoint has changed so much that the active patch warping fails to predict the current appearance.

Second, a consistency test based on the Mahalanobis distance (MD, see Appendix B) must hold. As the innovation 3σ ellipse corresponds exactly to the region at $MD \leq 3$, this test is implicit in the search region: if we have found the feature inside the ellipse then the MD condition automatically holds. However, depending on the scan procedures, the found pixel can effectively lie outside the 3σ region (the local scan, which is ZNCC guided, can eventually lead to such a situation). In such cases, or in cases where a different sigma-bound to that of the ellipses wants to be given, the MD test must be explicitly evaluated. This test will help to detect features that do not correspond to true 3D landmarks such as partially occluded edges or moving objects.

Finally, on failure to validation one has to consider a badly defined landmark. The ratio of successful matchings over total attempts is evaluated. If this ratio drops below 50% the landmark is considered unstable and deleted. Only successful measures will be used to update the SLAM map.

6.6 Experiments

6.6.1 Outdoors, the ‘Boxes’ experiment

Different experiments in real scenarios have been carried out. A representative one corresponds to that in Fig. 6.20, in which a vehicle (an all terrain rover) performs a straight trajectory that passes between the two boxes. The motion axis corresponds roughly to the central point of the horizon line in the image, slightly above the lower left corner of the round hut at the further end. Observability is very weak in the four image regions surrounding this point. It gets better as we move away from it. The bottom regions are perfectly observable, but the rapid changes in appearance difficult matching there, lowering feature stability. The sequence and the performing algorithm can be appreciated in movie format in `http://www.laas.fr/~jsola/objects/videos/PhD/video-6N.mov` , where `N` is a video number, or in the included CD.

The different parameters are tuned as indicated in Table 6.2 below. For the image regions for new initializations (Section 6.5.2), we take a grid of 6×6 regions and impose at least one feature at each one of the 4×4 inner regions. A maximum of 7 new initializations per frame is allowed, and the overall number of coexisting rays is limited to 12. The rest of parameters is clearly identifiable in the table.

Table 6.2: Parameter tuning for the 'Boxes' experiment.

Topic	Section	Parameters		Values	Comment
Geometric ray	6.4.1	$\{s_{min}, s_{max}, \alpha, \beta\}$	=	$\{1, 30, 0.3, 3\}$	$N_g = 4$
FIS	6.4.2	$\{\gamma, \mu, \tau\}$	=	$\{0.05, 1, 0.001\}$	
Landmark model	6.5.1	patch size	=	15×15	
Image regions	6.5.2	grid	=	6×6	inner 4×4
Number of rays	6.5.2	{new, max}	=	$\{7, 12\}$	
Information gain	6.5.3	$\{N_p, N_r\}$	=	$\{8, 10\}$	
Active search	6.5.3	Ellipses size	=	3σ	prob. = 99%
Meas. validation	6.5.3	$\{\text{ZNCC}_{min}, \text{MD}_{max}\}$	=	$\{0.9, 3\}$	

For this 3D experiment we disposed of the robot's 2D odometry, which we used to improve predictions and, more important, to fix the scale factor. We chose for it a very simple error model in which translation noise variances σ_x^2, σ_y^2 and σ_z^2 and rotation (the three Euler angles) noise variances σ_ϕ^2, σ_θ^2 and σ_ψ^2 are proportional to the performed forward displacement Δx:

$$\begin{aligned} \sigma_x^2 = \sigma_y^2 = \sigma_z^2 = {k_d}^2 \cdot \Delta x \\ \sigma_\phi^2 = \sigma_\theta^2 = \sigma_\psi^2 = {k_a}^2 \cdot \Delta x \end{aligned} \tag{6.40}$$

with $k_d = 0.04\text{m}/\sqrt{\text{m}}$ and $k_a = 0.02\text{rad}/\sqrt{\text{m}}$.

We used a camera with 512×384 pixel resolution and about 90° field of view (FOV) which provided gray-level images. As an accurate eye will recognize in the figures, for this experiment *distortion correction* was previously applied to the images (via a tabulated back-mapping).

The sequence consists of 97 images, taken at $\Delta x = 7$cm intervals approximately. Fig. 6.21 shows a snapshots sequence at 5 frames intervals. Observe how rays are initialized and their convergence to single points at the lower corners of both boxes (with fair observability) while at the upper ones (weak observability) the initialized rays remain as rays. They are used to improve camera (and robot!) localization thanks to the undelayed initialization.

The rays initialization and pruning based on reasoning in the image plane can be better appreciated in the zoomed sequence of consecutive snapshots shown in Fig. 6.22.

A systematic error of about 0.6°/m was perceived in the angular odometry readings: while the robot was slightly turning left, the integration of the odometry data results in a slow turn to the right. The 3D reconstruction did not suffer from this angular bias and the final estimated trajectory represents well the real one, *i.e.* it turns to the left (note that precise ground truth is not available).

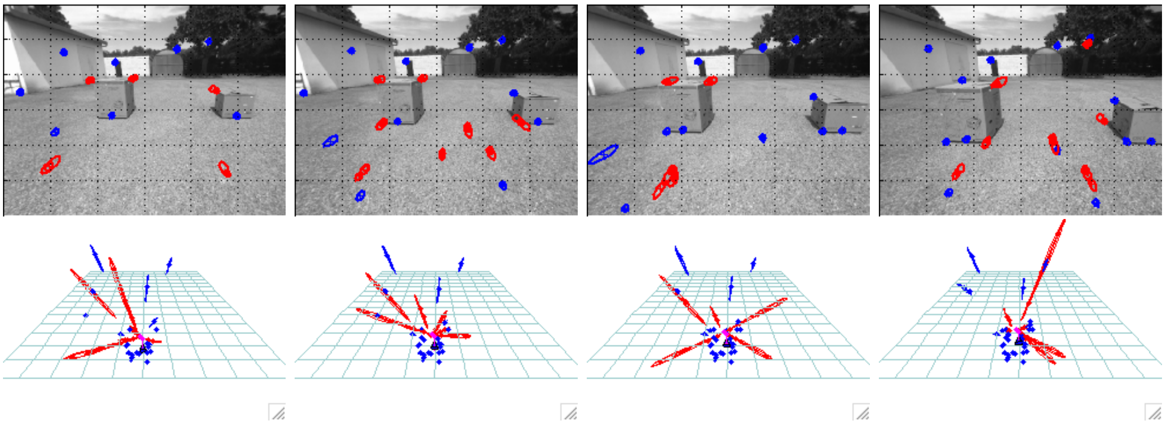

Figure 6.21: A portion of the reconstruction sequence at 5 frames (approx. 35cm) intervals. Image plane and 3D reconstruction are shown for frames 46, 51, 56 and 61.

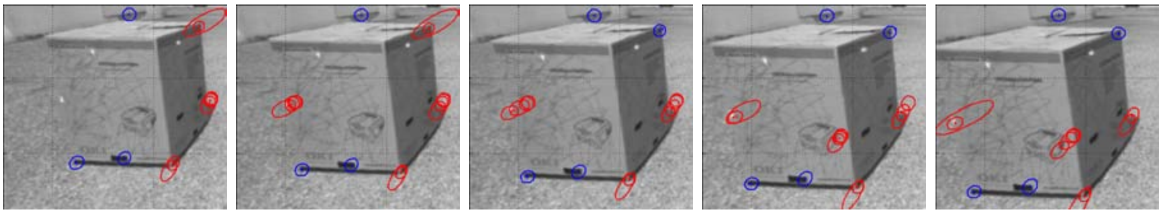

Figure 6.22: A zoom of another portion of the reconstruction sequence at 1 frame (approx. 7cm) intervals. Image plane is shown for frames 64 to 68.

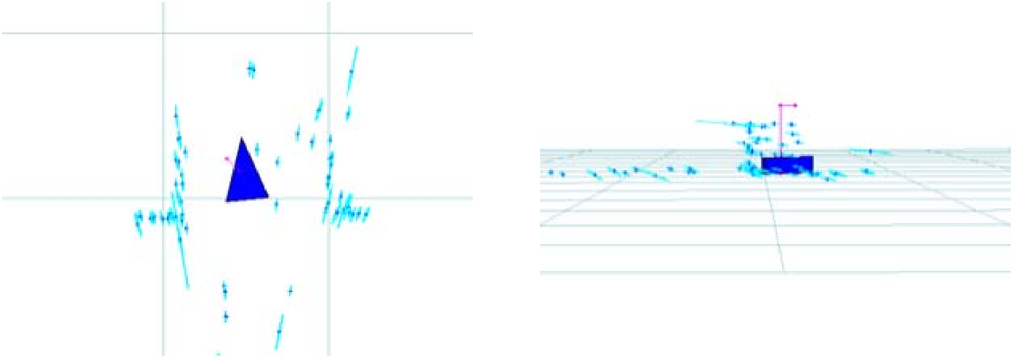

Figure 6.23: Top and side views of the reconstructed boxes at the end of the sequence.

The reconstructed structure of the two boxes is shown in Fig. 6.23. On the top view the structure of the boxes is clearly visible. On the side view, points on the ground are also well reconstructed. The recovery of the scale factor is shown to be correct as these ground points are effectively on the ground: they are consistent with the fact that the camera is at a precise height of 1.02m.

6.6.2 Indoors, the ‘White-board’ experiment

A second experiment with a longer series of distorted images has been performed indoors the robotics lab at LAAS (Fig. 6.24). The ‘White-board’ experiment is set up as follows: A robot

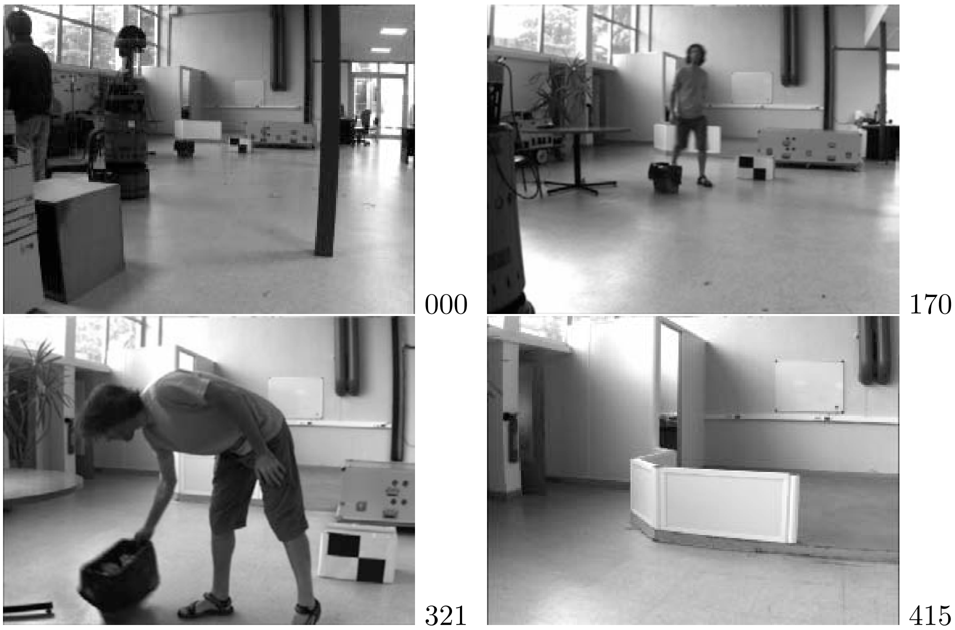

Figure 6.24: Four snapshots of the 'White-board' sequence inside the robotics lab at LAAS. The robot will approach the white-board at the end wall in a straight trajectory. We will notice the presence in the scene of a robot, a table, a bin, a small box, a trunk, a fence and, at the end, the white-board. We will also notice the presence of a special guest in motion which will produce occlusions and even the displacement of some objects.

with two cameras looking forward is run for some 15m in straight line towards the white-board at the end wall. Over 500 image-pairs are taken at approximatively 5Hz frequency, and the images exclusively from the left-hand camera are taken to feed the FIS-SLAM algorithms. The robot approaches the objects to be mapped, a situation that is common in mobile robotics but that presents observability difficulties for mono-vision SLAM because the trajectory is singular. The camera has 55° FOV at 512×384 pixels resolution, and has been intrinsically calibrated, with a radial distortion model of length 2 which has been inverted as explained in Chapter 2. A simple 2D odometry model similar to that in the 'Boxes' experiment is used for motion predictions.

The different parameters are tuned as indicated in Table 6.3. For the image regions for new initializations (Section 6.5.2), we take a grid of 7×7 regions and impose at least one feature at each one of the 5×5 inner regions. This will provide a greater feature density and hence a denser 3D map than in the 'Boxes' experiment. A maximum of 3 new initializations per frame is allowed, and the overall number of coexisting rays is limited to 15. The rest of parameters is clearly identifiable in the table.

The map after the whole sequence is shown in Figs. 6.25 and 6.26. It is difficult to see, from static 2D views, the 3D structure of a sparse set of points such as the map we built. An animated video of this sequence is available at the author's web page in http://www.laas.fr/~jsola/objects/videos/PhD/video-6N.mov , where N is a video number, or in the included CD.

The lengths of the four segments defining the white-board are compared to those in the real world in Table 6.4.

A second run with a much lower feature density (just a 4×4 regions grid) is performed to

Table 6.3: Parameter tuning for the 'White-board' experiment.

Topic	Section	Parameters		Values	Comment
Geometric ray	6.4.1	$\{s_{min}, s_{max}, \alpha, \beta\}$	=	$\{1, 15, 0.3, 3\}$	$N_g = 3$
FIS	6.4.2	$\{\gamma, \mu, \tau\}$	=	$\{0.05, 1, 0.001\}$	
Landmark model	6.5.1	patch size	=	15×15	
Image regions	6.5.2	grid	=	7×7	inner 5×5
Number of rays	6.5.2	{new, max}	=	$\{3, 15\}$	
Information gain	6.5.3	$\{N_p, N_r\}$	=	$\{7, 5\}$	
Active search	6.5.3	Ellipses size	=	3σ	prob. = 99%
Meas. validation	6.5.3	$\{\text{ZNCC}_{min}, \text{MD}_{max}\}$	=	$\{0.9, 3\}$	

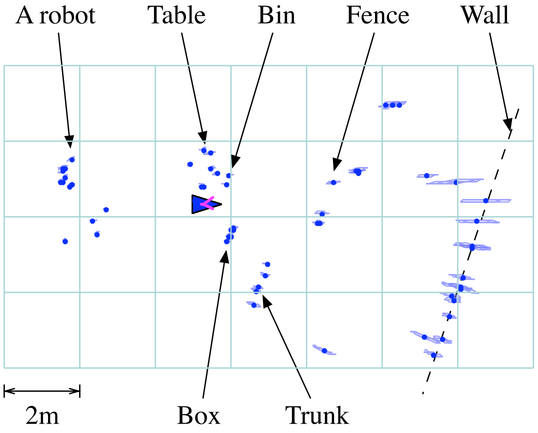

Figure 6.25: Top view of the map produced during the White-board experiment indicating the objects existing in the scene.

illustrate the undelayed initialization performance in Fig. 6.27. See how one of the hypotheses of the landmark on the floor (the closest landmark) had already attained the true value before the others got pruned. This is exclusive to undelayed methods where partially observed landmarks can effectively be corrected —and be used to get corrected.

6.7 Conclusions

The importance of undelayed initialization when performing monocular SLAM from a mobile vehicle has been highlighted as the only means to effectively consider the whole visual information, specially when the camera is looking forward, a situation that is found in the vast majority of cases in mobile robots and intelligent vehicles for obvious reasons. However, the implementation of such an undelayed initialization finds numerous difficulties that have to be overcome. We did this by mapping the whole ray derived from the first partial observation. The non-Gaussian character of this ray has led to a multi-hypothesis approximation which, in order to keep real-time constraints, has obliged us to develop a special filtering technique,

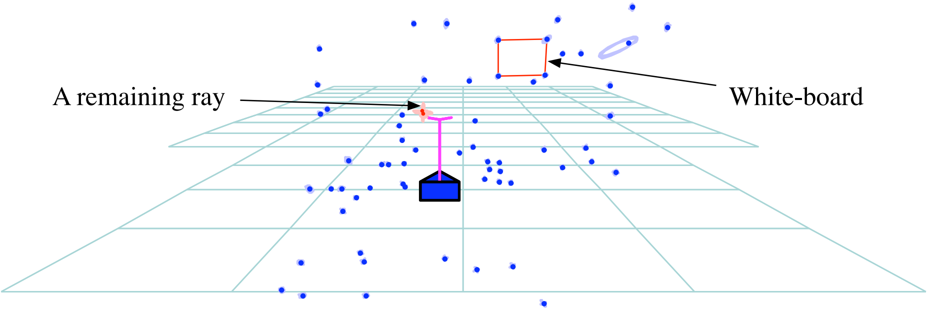

Figure 6.26: Robot-like view of the map produced during the White-board experiment.

Table 6.4: Map to ground truth comparison.

segment	location	real (cm)	mapped (cm)
A	board	116	115.4
B	board	86	82.6
C	board	116	110.5
D	board	86	81.2

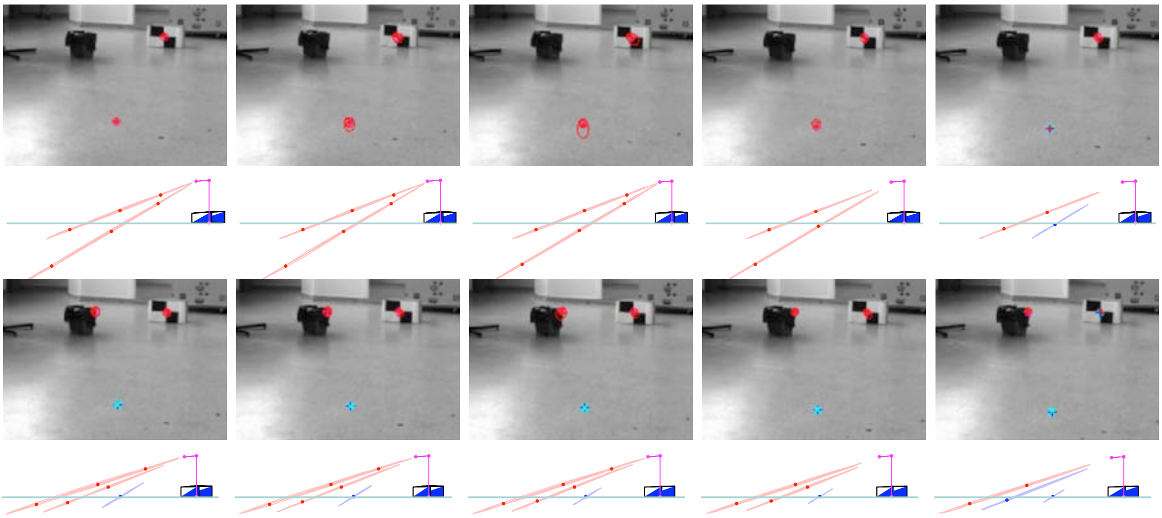

Figure 6.27: A detailed sequence during landmarks initializations in the 'White-board' experiment. One in every two frames is shown to appreciate the initialization and successive pruning, with the convergence of one hypothesis to the true landmark position. Rays are in red, single Gaussians in blue. Tiny dots indicate the Gaussian means.

that we named Federated Information Sharing, to host the SLAM operations. The visual information is extracted from the images in a robust and efficient way thanks to the active feature search approach which optimally exploits all the information contained in the SLAM map.

FIS is an approximated method to mimic the GSF operation with reduced algorithmic complexity. The drawbacks of this approximation have been highlighted and minimized, but convergence proofs have not been obtained and hence the trust we can have on the proposed methods comes exclusively from intuition and experiments. The algorithm proved to be quite robust for most of the monocular experiments performed, in our opinion because of the fact that, as successive camera positions are very close one another, all hypothesis of a ray significantly overlap when they are projected into the image. This means that all updates performed on the ray possess very small innovations, something that contributes to minimize the divergence and consistency risks mentioned.

When camera poses will be more distant the hypotheses innovations within a ray will become disparate and the filter operation could suffer from it. This situation will appear in the next chapter where rays initialized from one camera will be updated from another one. Therefore, we will have to talk about this again.

We strongly believe that this multi-hypothesized technique can now be substituted by the inverse-depth parametrization appeared recently in [Eade and Drummond 2006; Montiel et al. 2006], which can be more theoretically defended.[15] This does not lessen the merit of this work but, from today's new perspective, the principal contribution of this chapter remains, therefore, in the identification of the necessity of the undelayed methods and on its pioneering effort to open the track.

[15] However, the above comment on its performance when camera poses are more separate still holds in some way.

Chapter 7

Bi-camera SLAM

7.1 Introduction

We explore now the possibility of using the mono-vision algorithms in systems that possess more than one camera. The idea is to combine the advantages of both mono-vision (bearing-only, with infinity range but no 3D instantaneous information) and stereo-vision (3D information only up to a limited range) to build a system that is able to instantaneously map close objects while still considering the information provided by the observation of remote ones. The idea, in other words, is to provide the instantaneous 3D observability typical of a stereo system while avoiding the dense-fog and frontal-blindness effects highlighted in Chapter 6.

In the development of this chapter, two cameras in stereo configuration are considered (Fig. 7.1), although the proposed ideas should be naturally exportable to any other configuration as long as the cameras show overlapping fields of view. As the mechanisms used are closer to mono-vision than to stereo-vision, we explicitly differentiate from 'stereo' by naming such a system *bi-camera*, which is closer to the mono-camera term used in the previous chapter.

With bi-camera SLAM (BiCamSLAM) we obtain the following fundamental benefits: *1)* important objects for reactive navigation, which are close to the robot, are instantaneously mapped with stereo-like triangulation; *2)* good orientation localization is achieved with bearings-only measurements of remote landmarks; and *3)* updates can be performed on any landmark that is only visible from one camera. Additionally, because the information we seek

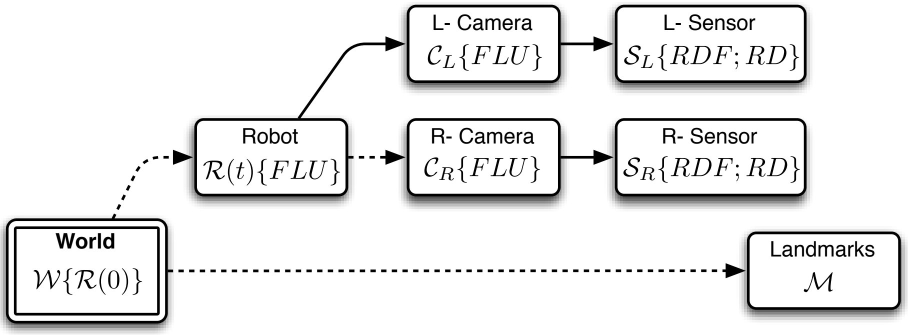

Figure 7.1: Reference frames diagram in BiCamSLAM (refer to Chapter 1 for notation). Solid-line arrows indicate deterministic definitions. Dashed-line ones are uncertain; their *pdfs* are continuously estimated by BiCamSLAM.

is sparse, we also obtain important technological benefits: *4)* no need for image rectification[1]; which allows us to say that *5)* precise previous calibration of the stereo rig extrinsic parameters of the cameras is no longer necessary; because *6)* dynamic self-calibration of these stereo rig extrinsic parameters can be incorporated, thus making such an intrinsically delicate sensor more robust and accurate. These two latter assertions are demonstrated with a simplified self-calibration procedure based on the same EKF used for SLAM. The key for all these benefits is using mono-vision algorithms in both cameras instead of a stereo one: we get enhanced observability with a much greater flexibility.

Combining both mono- and stereo-vision we get an instantaneous observability of close frontal objects while still utilizing the information of distant ones: the first beneficiary is the robot localization as we will dispose of long term absolute angular references. It is known that it is precisely the accumulation of angular errors due to uncertain robot motion which makes simple SLAM algorithms (such as EKF-SLAM) become inconsistent and fail [Castellanos et al. 2004]. Thus, this long term observability will improve EKF-SLAM performance.

This chapter covers the following topics. In Section 7.2 we detail a method to conditionally initialize landmarks in the SLAM map depending on whether they are fully observable or not. In Section 7.3 we present a simple method to self-calibrate the stereo rig extrinsic parameters. We add some comments and special details on visual perception in 7.4. We present some experiments in 7.5 and conclude in 7.6 with a brief discussion and indications for future work.

The research contained in this chapter led to the following publications: [Solà et al. 2006; Solà et al. 2007].

7.2 Landmarks initialization

Bi-camera vision *is not* stereo vision, in the sense that we are not explicitly observing distances (or depths, or disparities). It is just two times mono-vision that takes advantage of the enhanced observability that instantaneous ray triangulation provides, like stereo-vision does. The information enters the system always as bearings-only, and it is the filter the responsible of fusing it to obtain the 3D.

As a general idea, one can simply initialize landmarks following mono-vision techniques from the first camera, and then observe them from the second one: the fusion filter will determine their 3D positions with more or less accuracy depending on if these landmarks are located inside or outside the stereo observability region. But we can easily go one step forward: understanding the angular properties that generated these observability regions (as we did in Chapter 5, where we imposed the two external 4σ bounds of the rays to be parallel), we can *a-priori* evaluate, from both images, whether each landmark is fully 3D-observable or not. Landmarks will be initialized in a different fashion depending on the result of this evaluation (Fig. 7.2).

7.2.1 Observability evaluation

Consider two cameras in typical stereo configuration (parallel optical axes; coplanar image planes; camera optical centers in the same horizontal plane) and no distortion. A detailed

[1] Rectification in stereo simultaneously corrects distortion and re-maps images so that the epipolar line of a feature in one image is an horizontal line in the other one at the same height of the former. This allows for feature search only along the same line of pixels, obtaining fast and dense 3D reconstruction.

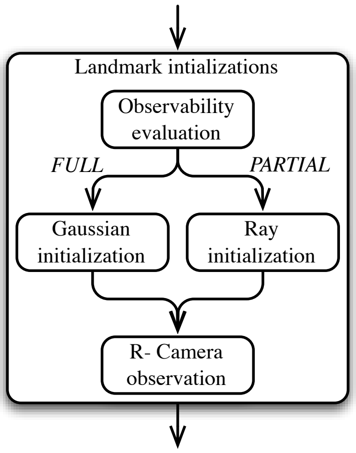

Figure 7.2: Conditional landmarks initialization in BiCamSLAM.

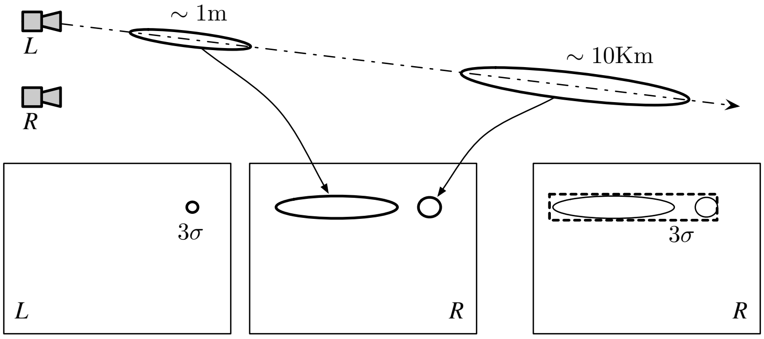

Figure 7.3: The 2-member ray in BiCam initialization. The left- and right- 3σ projections, and the search region in the right image.

description of the observability evaluation method is illustrated in Fig. 7.3 and described as follows:

Assume a new feature is detected in the left image. In the robot frame, the back-projection function from the left camera is simply $\mathbf{p} = \mathbf{g}(\mathcal{C}_L, s, \mathbf{b}_L)$ (Chapter 2), where $\mathcal{C}_L$ is the left camera frame, s is an unknown depth and $\mathbf{b}_L \sim \mathcal{N}\{\mathbf{y}_L; \mathbf{R}\}$ is the uncertain landmark projection centered at the position $\mathbf{y}_L$ of the detected feature. Define (without initializing it in the SLAM map) a 2-members ray in the robot frame: one member $\mathbf{p}_1 \sim \mathcal{N}\{\bar{\mathbf{p}}_1; \mathbf{P}_1\}$ is at the minimum considered depth s_1 and the other $\mathbf{p}_\infty \sim \mathcal{N}\{\bar{\mathbf{p}}_\infty; \mathbf{P}_\infty\}$ at the maximum, s_∞, virtually at infinity. These two members are easily specified following the directives of Section 6.4.2 in Chapter 6 as follows:

$$\begin{aligned}\bar{\mathbf{p}}_i &= \mathbf{g}(\mathcal{C}_L, s_i, \mathbf{y}_L)\\ \mathbf{P}_i &= \mathbf{G}_{\mathbf{b},i}\, \mathbf{R}\, \mathbf{G}_{\mathbf{b},i}^\top + \mathbf{G}_{s,i}\, \sigma_i^2\, \mathbf{G}_{s,i}\end{aligned}$$

where $i \in \{1, \infty\}$, $\sigma_i = \alpha\, s_i$ with α the shape factor of the geometric ray members, and $\mathbf{G}_\mathbf{b}$ and $\mathbf{G}_s$ the appropriate Jacobian matrices. Now project this ray onto the right image: the nearby member becomes an elongated ellipse; the remote one, that projects exactly at the vanishing point of the ray, is a rounded, smaller ellipse. The axis joining both ellipse centers is

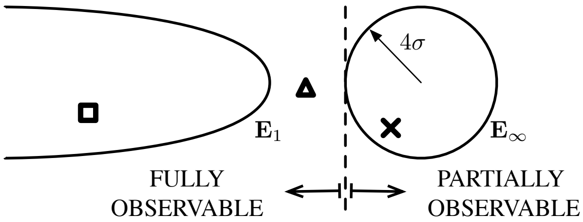

Figure 7.4: Deciding on 3D observability. A 4σ criterion is *a-priori* reasoned in the 2D image plane. The measure marked 'cross' corresponds to a landmark outside the stereo observability region. Landmarks measured 'square' and 'triangle' are inside.

precisely the epipolar line of the feature, and in the case we are considering it is an horizontal line. Let the Jacobian matrices of the right camera observation function $\mathbf{b}_R = \mathbf{h}(\mathcal{C}_R, \mathbf{p})$ with respect to the right camera pose $\mathcal{C}_R \sim \mathcal{N}\{\bar{\mathcal{C}}_R; \mathbf{P}_{\mathcal{C}_R}\}$ and the point positions $\mathbf{p}_1$ and $\mathbf{p}_\infty$ be defined as

$$\mathbf{H}_{\mathcal{C},1} = \left.\frac{\partial \mathbf{h}}{\partial {\mathcal{C}_R}^\top}\right|_{\bar{\mathcal{C}}_R, \bar{\mathbf{p}}_1} \quad \mathbf{H}_{\mathcal{C},\infty} = \left.\frac{\partial \mathbf{h}}{\partial {\mathcal{C}_R}^\top}\right|_{\bar{\mathcal{C}}_R, \bar{\mathbf{p}}_\infty} \quad \mathbf{H}_{\mathbf{p},1} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{p}^\top}\right|_{\bar{\mathcal{C}}_R, \bar{\mathbf{p}}_1} \quad \mathbf{H}_{\mathbf{p},\infty} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{p}^\top}\right|_{\bar{\mathcal{C}}_R, \bar{\mathbf{p}}_\infty}.$$

Let $\mathbf{R}$ be the covariances matrix of the right-hand camera measurements and $\mathbf{P}_{\mathcal{C}_R}$ that of this camera pose uncertainty. The projected $n\sigma$ ellipses correspond to the expectations $\mathbf{e}_i \sim \mathcal{N}\{\bar{\mathbf{e}}_i; \mathbf{E}_i\}$, $i \in \{1, \infty\}$ which are centered at their means

$$\bar{\mathbf{e}}_1 = \mathbf{h}(\bar{\mathcal{C}}_R, \bar{\mathbf{p}}_1) \tag{7.1}$$
$$\bar{\mathbf{e}}_\infty = \mathbf{h}(\bar{\mathcal{C}}_R, \bar{\mathbf{p}}_\infty) \tag{7.2}$$

and are described by their covariances matrices

$$\mathbf{E}_1 = \mathbf{H}_{\mathbf{p},1}\,\mathbf{P}_1\,\mathbf{H}_{\mathbf{p},1}^\top + \mathbf{H}_{\mathcal{C},1}\,\mathbf{P}_{\mathcal{C}_R}\,\mathbf{H}_{\mathcal{C},1}^\top + \mathbf{R} \tag{7.3}$$
$$\mathbf{E}_\infty = \mathbf{H}_{\mathbf{p},\infty}\,\mathbf{P}_\infty\,\mathbf{H}_{\mathbf{p},\infty}^\top + \mathbf{H}_{\mathcal{C},\infty}\,\mathbf{P}_{\mathcal{C}_R}\,\mathbf{H}_{\mathcal{C},\infty}^\top + \mathbf{R}. \tag{7.4}$$

Following the active feature search approach, the region including both 3σ ellipses is scanned for a feature match. The found pixel $\mathbf{y}_R$ is sent to the following 4σ observability evaluation test (Fig. 7.4), equivalent to that in Section 5.4:

Criterion 7.1 (The 4σ observability test):

The measured landmark is fully 3D observable if and only if the measured feature falls strictly at the left-hand side of the $\mathbf{e}_\infty$ *ellipse's leftmost* 4σ *border.* ◇

If we write the measured pixel $\mathbf{y}_R$ and the remote expectation $\mathbf{e}_\infty \sim \mathcal{N}\{\bar{\mathbf{e}}_\infty; \mathbf{E}_\infty\}$ as

$$\mathbf{y}_R = \begin{bmatrix} y_u \\ y_v \end{bmatrix} \qquad \bar{\mathbf{e}}_\infty = \begin{bmatrix} \bar{e}_{\infty,u} \\ \bar{e}_{\infty,v} \end{bmatrix} \qquad \mathbf{E}_\infty = \begin{bmatrix} \sigma^2_{\infty,u} & \sigma^2_{\infty,uv} \\ \sigma^2_{\infty,uv} & \sigma^2_{\infty,v} \end{bmatrix},$$

where $(\cdot)_u$ denotes horizontal coordinates, then this criterion resumes simply to

$$y_u < (\bar{e}_{\infty,u} - 4\,\sigma_{\infty,u}) \iff \text{FULLY 3D OBSERVABLE.} \tag{7.5}$$

In the presence of distortion or for a non-typical configuration of the stereo bench (*i.e.* two cameras in general configuration with overlapping fields of view), the same test must be

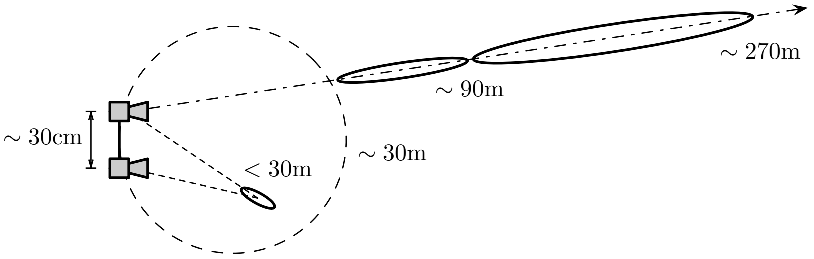

Figure 7.5: BiCam conditional initializations. Use stereo capabilities when possible. Use mono otherwise. When combined with self-calibration, get rays ranging hundreds of meters with very few members.

applied in the direction of the epipolar line (the line joining both ellipse centers) instead of the horizontal.[2] The test adaptation consists in a simple rotation of the image plane's coordinate frame. However, for the numerous cases of uncalibrated stereo rigs and typical radial distortions, the epipolar line is reasonably horizontal and the test can be applied as-is.[3]

7.2.2 Conditional landmark initializations

The landmark is then initialized either as a single point or as a ray as follows (Fig. 7.5):

1. If it is *fully observable*, it is clear that initializing the whole ray and then deleting all but the right members is not so clever. Better, we compute its depth by triangulation, and initialize a 'ray' of one single member at this depth using one of the views. We immediately update it with the second view to refine its position.

2. If it is *not fully observable*, a ray is initialized with its closest member already outside the region. The region limit in the ray direction is determined by triangulation with a virtual measurement at the critical point $\mathbf{y}_R^* = [y_u^*, y_v^*]^\top$ with $y_u^* = \bar{e}_{\infty,u} - 4\sigma_{\infty,u}$ and y_v^* chosen so that $\mathbf{y}_R^*$ lies at the epipolar line (Fig. 7.6), that is

$$\mathbf{y}_R^* = \bar{\mathbf{e}}_\infty - \frac{4\sigma_{\infty,u}}{\bar{e}_{1,u} - \bar{e}_{\infty,u}} \cdot (\bar{\mathbf{e}}_{1,u} - \bar{\mathbf{e}}_{\infty,u}). \tag{7.6}$$

   Triangulation of this critical point $\mathbf{y}_R^*$ with the observation from the left camera $\mathbf{y}_L$ (the one we used to define the ray) gives the minimum depth s_{min} which is used to define the ray's members parameters (Chapter 6). As the farther member's depths follow a geometric series, we easily reach ranges of several hundred meters with very few members. Once initialized, the ray is immediately updated with the observation from the other camera.

7.3 Stereo rig self-calibration

Stereo rigs are mechanically delicate, specially for big base lines. We believe that stereo assemblies are only practical if they are very small or if their main extrinsic parameters are

[2] Notice also that in the presence of lens distortion the epipolar line is not a straight line. This can be however neglected as the test is just a probabilistic test and hence approximations with straight lines are reasonable.

[3] Also, if we feel insecure with this, the same test can be made more conservative by just taking a 5σ bound instead of the 4σ proposed (at the price of reducing the full observability region size).

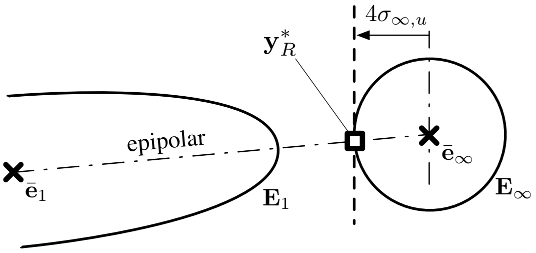

Figure 7.6: The critical point $\mathbf{y}_R^*$ that is used to determine the observability region limit in the ray direction via triangulation.

continuously self-calibrated. Outdoors operation will often impose this second case, therefore making self-calibration a *desirable* capability.

Notice that the observability evaluation method above inherently accounts for arbitrary extrinsic parameters accuracies: the size of $\mathbf{E}_\infty$ will vary accordingly, and hence the stereo-observable region bounds too. This means that we can cope with a dynamic behavior of these extrinsic parameters and accuracies, hence making self-calibration *well adapted*.

Notice also that, unlike stereo which only obtains three independent measurements per landmark (two image coordinates plus disparity), with BiCam we dispose of four of such measures (two image coordinates per image). This extra information enhances system observability and will actually make extrinsic self-calibration *possible*.

Notice finally that, by using the mono-camera SLAM solutions, we are able to obtain the localizations of both cameras with respect to the world. If we coordinate perceptions on both sides and unify the world representation we guarantee that the world they modeled is the same. Thus cameras are localized with respect to the same world, which means that we can recover the localization of one camera with respect to the other one. That is to say, *we have the means* to perform extrinsic self-calibration.

And now the unavoidable conclusion: if something that is ‘desirable’ and ‘well adapted’ is as-well ‘possible’ and we realize that ‘we have the means’ to get it, then we must run and get it. This is what we examine now.

Not all six extrinsic parameters (three for translation, three for orientation) need to be calibrated. In fact, the notion of *self-calibration* inherently requires the system to possess its own gauge. In our case, the metric dimensions or *scale factor* of the whole world-robot system can only be obtained either from the stereo rig base line, which is one of the extrinsic parameters (and notice that then it is absurd to self-calibrate the gauge!), or from the odometry sensors, which often are much less accurate than any rude measurement we could make of this base line. Additionally, as cameras are actually angular sensors, vision measurements are much more sensible to the cameras orientations than to any translation parameter. This means that vision measurements will contain little information about these translation parameters. In consequence, self-calibration should concern only orientation, and more precisely, the orientation of the right camera with respect to the left one. The error of the overall scale factor will mainly be the relative error we did when measuring the rig's base line.

We have used a very simple self-calibration solution which has given promising results: we just add three angles (or any other orientation representation we are familiar with —we actually use the quaternion) to the EKF-SLAM state vector (not forgetting the Jacobians of all involved functions with respect to them) and let EKF make the rest. The map gets the

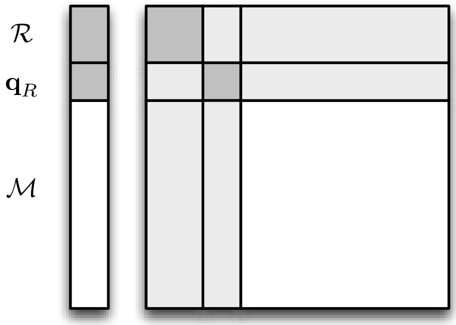

Figure 7.7: The BiCamSLAM map with extrinsic self-calibration. The quaternion $\mathbf{q}_R$ encoding the orientation of the right-hand camera is stacked in the state vector and jointly estimated by the EKF.

aspect shown in Fig. 7.7. The SLAM state vector becomes

$$X = \begin{bmatrix} \mathcal{R} \\ \mathbf{q}_R \\ \mathcal{M} \end{bmatrix}$$

where $\mathcal{R}$ and $\mathcal{M}$ form the previous robot and landmarks map, and $\mathbf{q}_R$ is the orientation part of the right-hand camera frame $\mathcal{C}_R$. The time-evolution function of the extrinsic parameters is simply $\mathbf{q}_R^+ = \mathbf{q}_R + \gamma$, where γ is a white, Gaussian, low energy process noise that accounts for eventual de-calibrations (due to vibrations or the like). For short-duration experiments we set $\gamma = 0$. The initial uncertainty will be set from a coarse analysis of the stereo structure's mechanical precision, and will generally be of the order of one or two degrees per axis. This can be reduced to several tenths of degree in cases where we dispose of previous calibrated values about which we are not confident anymore.

Although it works pretty well, this solution lacks some robustness and is included here as an illustration of the BiCam capability of working with on-line extrinsic calibration. This fact –this lack of robustness– is observed during the experiments and further discussed in Section 7.6.

7.4 Updates

Thanks to the mono-vision formulation, updates can be performed at any mono-observation of landmarks. This includes any nearby or remote landmark that is visible from both or from only one camera.

As indicated in the previous chapter (Section 6.5.3), the determinant of the expectation's covariances matrix is a measure of the information we will gain when measuring a particular landmark. This is so because the uncertainty in the measurement space can be associated to the surface of the corresponding ellipse, which is proportional to the square root of this determinant. Therefore, we suggest as a first step to organize all candidates to be updated in descending order of expectation determinants, without caring if they are points or rays, or in the left- or right- image, and update at each frame a predefined number of them (usually around 20). A second group of updates should be performed on remote landmarks (points or rays) to minimize the angular drift. Updates are processed sequentially, with all Jacobians being each time re-calculated to minimize the effect of linearization errors.

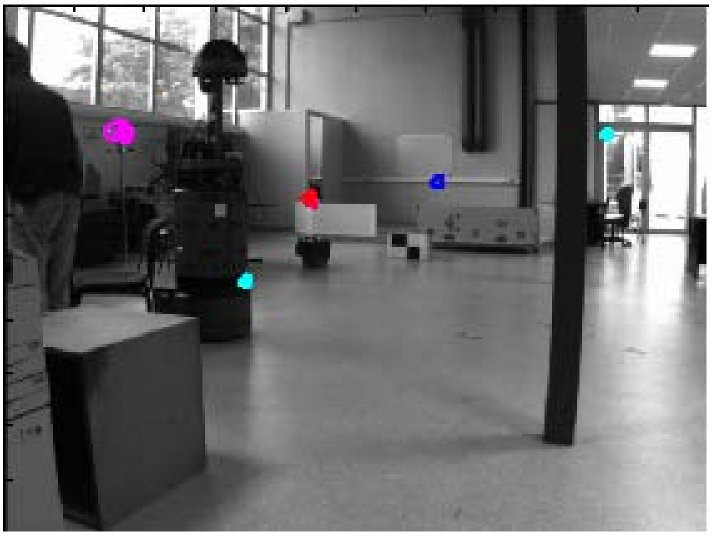

Figure 7.8: The LAAS robotics lab. The robot will approach the scene in a straight forward trajectory.

7.5 Experiments

The 'White-board' indoor experiment is resumed here in BiCam configuration to illustrate the proposed ideas. A robot with a stereo head looking forward is run for some 15m in straight line inside the robotics lab at LAAS (Fig. 7.8). Over 500 image-pairs are taken at approximatively 5Hz frequency. The robot approaches the objects to be mapped, a situation that is common in mobile robotics but that presents observability difficulties for mono-vision SLAM because of the singular trajectory. The stereo rig consists of two intrinsically calibrated cameras arranged as indicated in Table 7.1. The left camera is taken as reference, thus deterministically specified, and the orientation of the right one is initialized with an uncertainty of 1° standard deviation. A simple 2D odometry model is used for motion predictions. This experiment shows *a)* the self-calibration procedure; *b)* the initialization mechanism where the landmarks can be mapped with either a single Gaussian or a ray depending on the current 3D observability; and *c)* the metric accuracy of the resulting map. Illustrating videos can be found on the author's web page at `http://www.laas.fr/~jsola/objects/videos/PhD/video-6N.mov` , where `N` is a video number, or in the included CD.

Table 7.1: Stereo rig parameters in the 'White-board' experiment.

Scope	Parameters		Values
Dimensions	Base line	=	330mm
Orientation	{pan, tilt}	=	$\{0°, -5°\}$
Orientation - Euler	$\{\phi, \theta, \psi\}$	=	$\{0°, 5°, 0°\}$
Cameras	{resolution,FOV}	=	$\{512 \times 384\text{pix}, 55°\}$
Right camera uncertainties	$\{\sigma_\phi, \sigma_\theta, \sigma_\psi\}$	=	$\{1°, 1°, 1°\}$

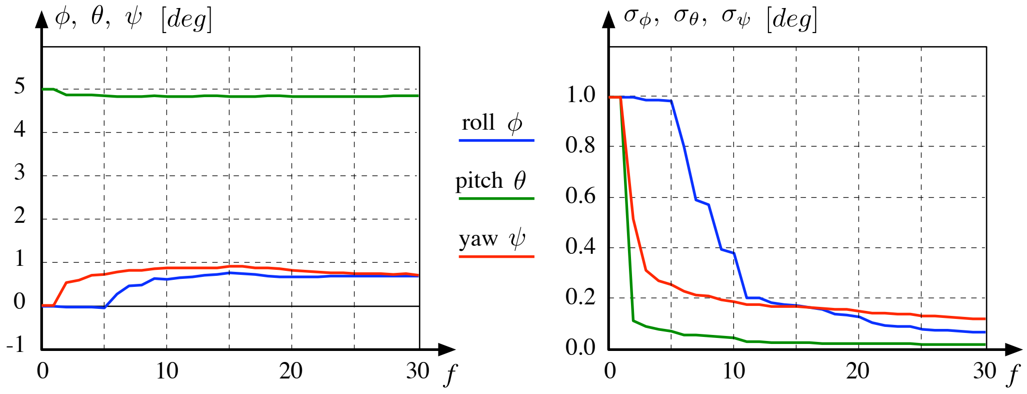

Figure 7.9: Extrinsic self-calibration. *Left:* The three Euler angles of the right camera orientation with respect to the robot as a function of the frame number during the first 30 frames. The nominal, initial values are $\{\phi_0, \theta_0, \psi_0\} = \{0°, 5°, 0°\}$; true values are $\{0.61°, 4.74°, 0.51°\}$. *Right:* Evolution of the standard deviations, also in degrees.

Self-calibration

The orientation of the right camera is specified with respect to the robot frame. A typical evolution, during the first frames, of the three Euler angles representation of the self-calibrated quaternion is illustrated in Fig. 7.9. We observe the following behavior:

- The pitch angle (cameras tilt, 5° nominal value) is observable from the first matched landmark. It rapidly converges to an angle of 4.87° and remains very stable during the whole experiment.

- Roll angle is observable after at least two landmarks are observed. It may take some frames for this condition to arrive (here we purposely limited the number of initializations in order to exaggerate this effect) but then it also converges relatively fast and quite stably.

- Yaw angle is very weakly observable because it is coupled with the distance to the landmarks: both yaw angle and landmark depth variations produce a similar effect in the right image, *i.e.* the feature moves following the landmark's epipolar line. For this reason, it does start converging from the first initial uncertainty, but after some frames it does it insecurely and slowly: see how from frame 15 onwards yaw uncertainty is already bigger than roll one, which started converging later. As it can be appreciated in Fig. 7.10 the value of the estimated yaw angle only shows reasonable convergence after 150 frames but it is not very stable during the experiment length neither very repeatable among different experiments. Before these 150 frames, yaw estimates are clearly inconsistent: its true standard deviation, which can be appreciated in Fig. 7.10 to be about 1°, is much larger than its estimated value, which from Fig. 7.9 is about 0.1° at frame 30.

To conclude, we made 10 runs of 200 frames and collected the estimated calibration angles and standard deviations at the end of each sequence. We computed the statistical standard deviations (with respect to the 10 runs) of these estimated angles. We compared these values against the angles provided by the Matlab calibration toolbox. Apart from the mentioned initial stability issues, the results in Table 7.2 show a surprisingly good calibration, with

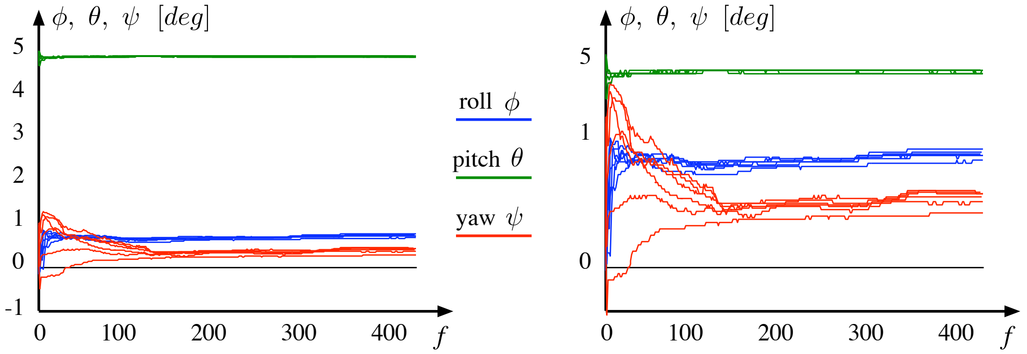

Figure 7.10: Calibration stability and repeatability. The plots corresponding to six diffent runs are overlapped to show repeatability. Pitch angle (*green*) is immediately and repeatably calibrated. Roll angle (*blue*) must wait until at least two points are present but shows good stability. Yaw angle (*red*) shows large variations in time and also between runs, although it seems to converge always to the same values. A 300% zoomed version of roll, pitch (not to scale) and yaw plots is shown on the right to better compare each angle's dispersions.

Table 7.2: Self-calibration errors with respect to off-line calibration.

Euler angle	off-line calib.	self-calib.	error	σ (statistical)	σ (estimated)
roll ϕ	0.61°	0.60°	−0.01°	0.038°	0.021°
pitch θ	4.74°	4.87°	0.13°	0.006°	0.006°
yaw ψ	0.51°	0.33°	−0.18°	0.108°	0.018°

similar statistical and estimated standard deviations, except for yaw which shows a clear inconsistency, *i.e.* an overestimate of its standard deviation.

Initialization mechanism

We show now, with a different sequence, the dynamic observability decision criterion with extrinsic self-calibration. The operation during the first three frames is detailed in the three columns of Fig. 7.11. On the top row pictures (left camera images), the tiny green ellipses are the projections of the 2-member ray. On the second row (right camera images) we see the 3σ search ellipses of expectations $\mathbf{e}_1$ and $\mathbf{e}_\infty$. On the third row, the initialized landmark is projected onto the right image plane: the corresponding ellipses are plotted in red for rays- and in blue for single-gaussian landmarks. Observe how, on the first frame, extrinsic self-calibration is poor and results in big decision ellipses (decison ellipses are 4σ, thus 33% bigger than the plotted search ones), giving place to initializations of nearby landmarks in the form of rays. Observations from the right camera refine the extrinsic precision and subsequent decision ellipses become smaller. On the third frame, the stereo rig is already quite accurate and is able to fully observe the 3D position of new landmarks. Previous rays are continuously observed from both cameras and will rapidly converge to single Gaussians as self-calibration enhances accuracy.

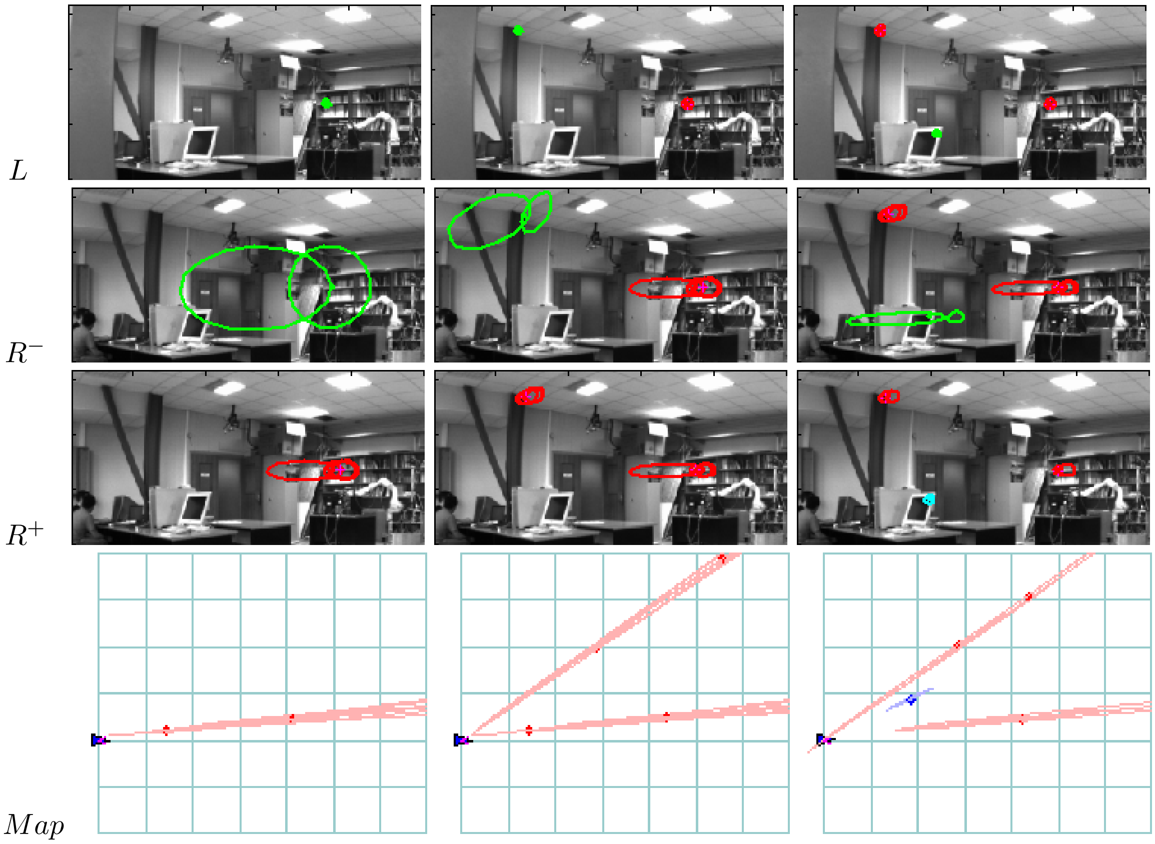

Figure 7.11: Initialization sequence in the presence of extrinsic self-calibration. The initialization sequence for the first three frames, one per column, is shown. Start at frame 1: a 2-member ray (*green*) is defined from the left view (*L row*). It is projected onto the right image (R^- *row*). The two 3σ ellipses (*green*) define a region which is scanned for a feature match. If this match is not on the left of the right-hand 4σ ellipse (33% bigger than drawn), the landmark is not 3D observable and is initialized as a ray (*red,* R^+ *row*). The resulting map is shown (Map row, the grid at 2m spacing). Subsequent observations (*columns 2 and 3*) increase calibration accuracy and hence the ellipses shrink. After 3 frames a newly detected landmark at a similar range is already 3D observable, thanks to the enhanced extrinsic precision, and can be initialized as a single Gaussian (*blue*).

Metric accuracy

Similarly to what we did in the mono-SLAM case, we show in Fig. 7.12 the top-view map of the LAAS robotics lab generated during this experiment. We will see no appreciable differences with respect to the mono-SLAM one (Fig. 6.25 on page 126). In fact, this is more a flattery to mono-SLAM than to BiCam because of its poorer 3D observability, but this is the way things go sometimes.

To contrast the resulting map against reality, two additional tests are performed: planarity and metric scale (Fig. 7.13). *1)* The four corners of the white board are taken together with 9 other points at the end wall to test co-planarity: the mapped points are found to be coplanar within 4.9cm of standard deviation error. *2)* The lengths of the real and mapped segments marked in red in Fig. 7.13 are summarized in Table 7.3. The white board has a physical size of 120 × 90cm but we take real measures from the approximated corners where the features are detected as shown in the figure. We observe errors in the order of one centimeter for

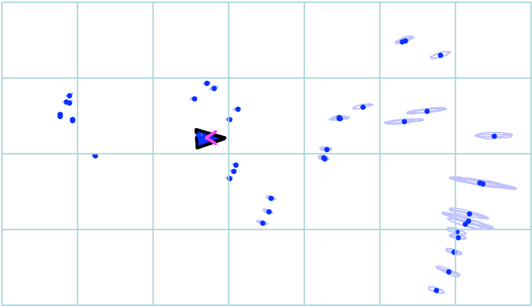

Figure 7.12: The map produced by the robot using BiCamSLAM. You are invited to compare it against the FIS-SLAM map on page 126.

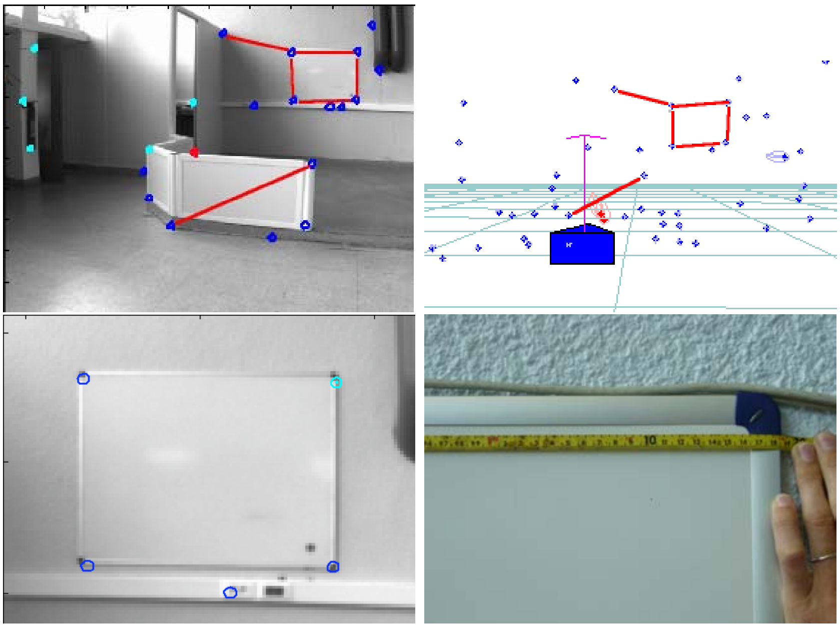

Figure 7.13: Metric mapping. The magnitudes of some segments in the real lab are compared to those in the map (red lines). The points defining the segments on the end wall are tested for co-planarity.

landmarks that are still about 4m away from the robot.

7.6 Conclusion and future work

We showed that using mono-vision SLAM techniques in multi-camera equipped robots provides several advantages. These advantages have been highlighted and explored with the FIS-SLAM algorithm, although they should come up naturally in any other implementation —and we think about single-hypothesis ones based on inverse-depth parametrization [Montiel et al. 2006; Eade and Drummond 2006].

The self-calibration solution proposed here suffers from poor observability and inconsistency problems. From the three angles to be calibrated, only roll (frontal axis rotation) and pitch (lateral axis) are strongly observable; yaw (vertical axis, this is the convergence angle of both cameras) is coupled with the distance to the landmarks. This is traduced into a strong

Table 7.3: Map to ground truth comparison.

segment	location	real (cm)	mapped (cm)
A	board	119	119.6
B	board	86	84.3
C	board	115	114.8
D	board	88	89.0
E	wall	134	132.5
F	fence	125	124.5

cross-correlation between them, and thus drifts in the map produce drifts in this angle and vice-versa. Theoretically speaking, this coupling should not be a problem as we know from multiple-view geometry [Hartley and Zisserman 2000] that an image pair of five 3D-points in general configuration renders the whole system observable, but things are in practice much more delicate. Regarding inconsistency and as it has been mentioned in the conclusion of Chapter 6, the fact of the different ray members being projected from one camera to the other one has some undesired implications in the case we are considering: the Principle of Measurement Reproduction used to inspire the FIS update method (Chapter 6) may give overestimate values in the direction where expectations are more disperse, and this is precisely the direction that couples the cameras convergence angle with the distance to the landmarks. We support this assertion with experimental evidence: the yaw angle should not start to converge before at least five points are mapped, but it actually does as we have seen in Fig. 7.9. If this suspicion about the FIS method were true, the adoption of the inverse-depth parametrization should notably improve self-calibration performance. In any case, further work must be done to insure a consistent, real-time, continuous calibration operation, eventually de-coupled from the main EKF filter, thus allowing the use of more robust methods running at a slower rate than the video stream.

Nevertheless, this procedure helped to prove with real experiments that, given a dynamic and uncertain set of extrinsic parameters, the 3D observability can be determined from very simple reasoning on the image plane. Of course one can use the whole BiCam proposals with an offline-calibrated stereo rig.

Chapter 8

Vision based SLAM with Moving Objects Tracking

8.1 Introduction

In this chapter we solve the SLAM with Moving Objects Tracking problem (SLAM-MOT) with the aid of vision. The goal is to achieve a convenient description of the visually perceived dynamic scene: how the static world is, where I am in this world, and how other existing bodies move. The SLAM-MOT project is undertaken in two different steps: a detailed observability evaluation, and a definition of convenient detection and tracking methods.

Regarding observability, we need to consider whether the problem is solvable from a mono-vision-equipped moving platform. This question admits two valid answers: a hopeful 'yes' and a deceptive 'no'. We will see under which conditions we can take the affirmative answer and compare them against the possibilities of a monocular SLAM approach such as that developed in Chapter 6. We will see that important additional work on image processing and/or multi-body 3D reconstruction would be needed to fulfill these conditions. Alternatively, and admitting that we are not ready to undertake a monocular solution, the possibility of using full 3D observability of the BiCam algorithm developed in Chapter 7 will lead to a straightforward tracking solution, which will obviously restrict MOT operation to the nearby region of full 3D observability, while using the whole (infinity range) visual field for SLAM.

Regarding the necessary methods it is convenient to formulate the following remark: while in the chapter's title we speak of moving objects tracking as a whole, we notice that the methods we need to conceive must in fact solve two problems of radically different nature: *Moving Objects Detection* (MOD) and *Moving Objects Tracking* (MOT). We will start by the second one, which is easier.

Indeed, the MOT problem is the part of the problem which is solved by filtering techniques. We will see that the tracking problem does not require much additional material. By accurately retaking all the material developed in the past chapters we will be able to set up an algorithm which satisfactorily fulfills our requirements. This algorithm will be described in detail.

We will see how, much more difficult than *tracking*, the MOD problem of initially *detecting* moving objects is the one that needs special attention. This difficulty arises from the impossibility of the machine to exhaustively analyze the whole surface of the images at every single frame. In fact, as each image is itself a static element, even with exhaustive scans this detection is not trivial. We present a two-step method to tackle this difficulty that, with a similar objective as active feature search, exploits the knowledge we have of the system to *anticipate* where potential moving objects may appear in the image, thus focusing the detecting procedures to those areas.

The chapter is organized as follows. In Section 8.2 we analyze some precedent approaches to the problem. In Section 8.3 we revise some observability issues for the mono-vision and multi-vision alternatives to justify the adoption of our BiCam method as the operating framework. In Section 8.4 we reorganize all the previously developed material to tackle the mobiles tracking and its insertion in the BiCam algorithm. In Section 8.5 we address the perception issues devoted to the problem of feature detection and to deciding whether detected features should be treated as moving objects or static landmarks. We present some preliminary results in Section 8.6 and conclude in 8.7 with a discussion.

The research contained in this chapter has not been published yet.

8.2 Precedents and our general position

The first remarkable work on SLAM-MOT is due to Wang [2004]. The system takes 2D laser-range-scanner based SLAM and incorporates tracking of moving objects. These objects are detected by isolating the portions of the current laser scan that do not show sufficiently good match with respect to the current SLAM map. Being useless as self-localization references, moving objects are maintained out of the map. The whole system relies on high speed, long range, laser range-finders. The most impressive aspect resides in its ability to perform large area SLAM at high speeds with long loop-closings in urban, dense-traffic conditions.

An alternative, non-SLAM based interesting approach is due to Agrawal et al. [2005]. Dense *disparity images* are produced by means of a calibrated stereo rig. A robust RANSAC method [Fischler and Bolles 1981] on triplets of matched points between two disparity images is used to determine the main camera-to-scene rigid transformation which is assigned to the camera motion. A *homography* is computed for each pixel based in this motion. Then the appearance of each pixel is compared to the appearance it should have based on the previous image and the predicted homography. Pixels that show sufficiently large appearance variation are classified as candidates to have suffered independent rigid motion, hence belonging to moving objects. Some blob grouping and spurious rejection is performed and finally a Kalman filter in 3D space is set for each moving object, with a constant-speed model, to obtain a reasonably sound trajectory and motion estimation. The system is able to turn at 16Hz on 320×240 images thanks to an accurate programming that exploits microprocessor-specific capabilities.

A third, more theoretically speaking approach to this problematic is due to Vidal et al. [2002a][2002b]. They formulate the question of whether one can detect and specify an unknown number of independent rigid motions given just two groups of matched features between two images. By an accurate analysis of a formulation of the *epipolar constraint* extended to the multi-body case, they are able to give the following answers: *a)* how many objects (independent rigid motions) there are; *b)* their motions and shapes (their respective structures and *fundamental matrices* —translation and rotation with respect to the camera up to an unrecoverable scale factor); and c) which point corresponds to which object. Additionally, *d)* most of these operations can be solved linearly (*i.e.* by means of just linear-algebra algorithms). While the mathematical soundness of this approach is solid and appealing, a real-time implementation of the derived algorithms has to tackle the $O(n^6)$ combinatorial complexity on the number n of moving objects.

These are the approaches we have notice about. Clearly enough, only the first is SLAM-based, *i.e.* tackled in a recursive filtering manner which is additionally able to produce a static reference map that allows re-localization after large loops, something that is not naturally included in the other memory-less approaches. As far as we are concerned, with all the material we have developed so far, this is the position we adopt to undertake our way into the problem. Apart from this fundamental position, the ideas that follow are not inspired by [Wang 2004] although the particularities of the problem itself will make our solutions, in the filtering aspects, very similar. Our method adds full 3D operation and visual perception, with all their inherent advantages —and difficulties.

8.3 Observability analysis

Before attempting the solution to the MOT problem we study here some observability issues. The conclusions of this study will determine the structure of our algorithm and the required sensing hardware —the number of cameras. The question is wether the trajectory of a randomly moving object is observable from a bearings-only platform or, on the contrary, full 3D sensing capabilities are needed.

8.3.1 Bearings-only sensing

The answer to this question in the bearings-only case depends on what we consider as being an 'object': a *point in space*; or a *rigidly linked set of points*. These two cases are separately explored in the next paragraphs.

Objects are points

When the object is defined as a point in space, the problem resumes to the well known 'bearings-only tracking' problem, where a moving platform equipped with a bearings-only sensor needs to estimate the trajectory of a punctual target. In such conditions, multiple relative observer-target trajectories can lead to exactly the same set of measurements, thus to non-observability.

A detailed observability analysis for this problem can be found in [Le Cadre and Jauffret 1997], where different conditions for both target and observer trajectories are considered. The conclusions are collected in Table 8.1 and may be (very informally) resumed with the assertion: "*The observer needs to perform more maneuvers than the target*" which, when the target maneuvers are unknown, translates to: "*The observer should move quite randomly, with faster dynamics than those of the target.*"

Notice that the target's and observer's constant speed assumptions, with only few maneuvers, constitute reasonable hypotheses only for very particular scenarios such as ocean navigation, with ships masses of thousands of tones (both for targets and observers). Notice additionally that in such a scenario (the immensity of the sea surface) the observer finds no constraints to choose those maneuvers that will permit him to maximize observability.

Robotics scenarios constitute an extremely opposite situation: 'targets' become 'objects', which in turn represent a diverse amalgam of everyday mobiles such as pedestrians, automobiles, other robots, etc. which may exhibit random-like behavior (like the dog of that old lady when you are on your bike). Additionally, observer trajectories are often highly constrained

Table 8.1: Bearings-only tracking. Trajectory conditions for observability.

Target speed	Maneuvers	Observer motion	Observable
constant	NO	constant speed	NO
constant	NO	one maneuver	YES
constant	N maneuvers	N+1 maneuvers (one per segment)	YES
variable	random	N maneuvers	NO
variable	random	random (independent from target)	YES

by both a structured environment (road, path, railway, static or moving obstacles) and by the presence of other fragile agents (delicate objects, other robots, nearby people) which might get injured or alarmed by such an unpredictable behavior. It is therefore highly inadvisable to use a bearings-only solution to track single points.

Objects are rigid structures of points

When the object is a structured set of points which are rigidly linked one another, the additional constraints arising from these links will provide the necessary material to render the system observable.

In fact, as we illustrate in Fig. 8.1, we can see each object-sensor pair as an independent problem where the object frame acts as the reference frame and the points of its structure are the landmarks. Each problem is solvable by different means (bearings-only SLAM as in Chapter 6, Bundle Adjustment from Structure From Motion, Eight Points Algorithm from Multiple View Geometry and maybe others) up to an unknown scale factor. Once solved, all sensor motions are imposed to be exactly the same: this operation fixes all reference frames with respect to a unique reference and, more important and remarkably, leaves a single unknown scale factor which is common to all solutions [Ma et al. 2004]. Odometry data or any other metric measurement may be used to determine this global scale.

Such kind of approach has three main disadvantages:

1. The necessity of *previous feature association*: we need to know, before the reconstruction, which features correspond to which objects. The only solid means to overcome this difficulty is the aforementioned work by Vidal et al. [2002a], but the combinatorial complexity of this part of the problem is prohibitive as stated. Alternatives could arise from RANSAC approaches (also with high combinatorial complexities) or from previous *segmentation* of the images (we mean color segmentation or the like) which could define initial guesses of feature-object associations, thus reducing the dimensionality of the initial search space. Additionally, every object must have enough number of points for the algorithms to work (this normally resumes to at least ten points per object).

2. The need for sufficient *numerical conditioning*, in a double sense: *a)* the different viewpoints must give enough angular aperture to render the reconstruction of any use; *b)* the apparent sizes of objects in the image must be relatively large in order to make the relative pixel error sufficiently small (and to contain at least ten features as mentioned). This condition is far from being fulfilled in most situations.

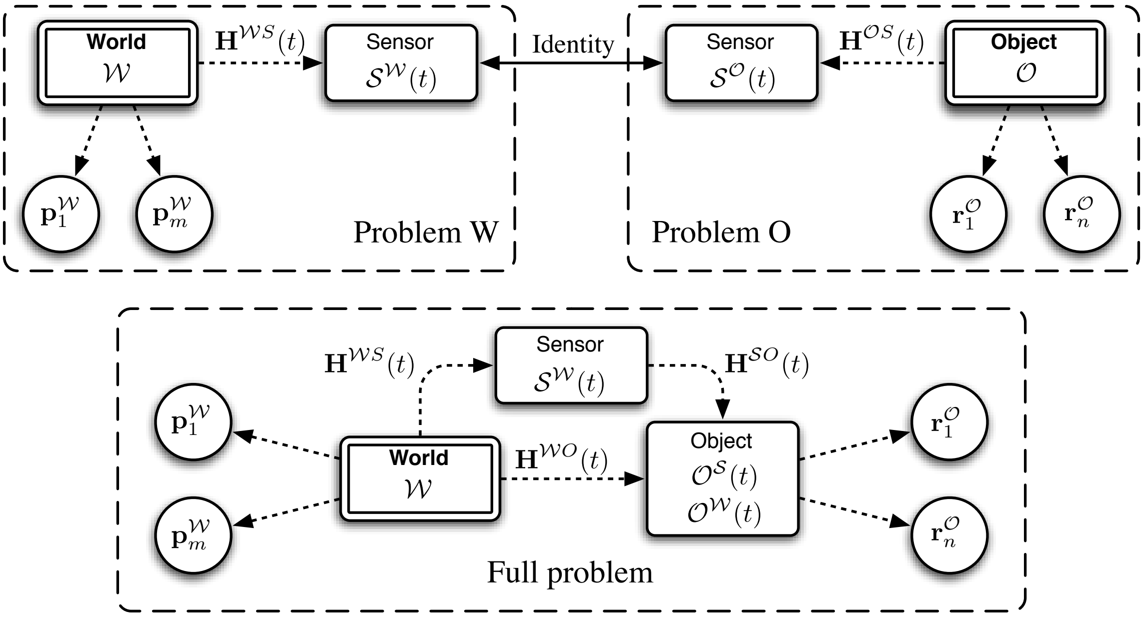
World
$\mathcal{W}$
$\mathbf{H}^{\mathcal{WS}}(t)$
Sensor
$\mathcal{S}^{\mathcal{W}}(t)$
Identity
Sensor
$\mathcal{S}^{\mathcal{O}}(t)$
$\mathbf{H}^{\mathcal{OS}}(t)$
Object
$\mathcal{O}$
$\mathbf{p}_1^{\mathcal{W}}$
$\mathbf{p}_m^{\mathcal{W}}$
Problem W
Problem O
$\mathbf{r}_1^{\mathcal{O}}$
$\mathbf{r}_n^{\mathcal{O}}$
$\mathbf{H}^{\mathcal{WS}}(t)$
Sensor
$\mathcal{S}^{\mathcal{W}}(t)$
$\mathbf{H}^{\mathcal{SO}}(t)$
$\mathbf{p}_1^{\mathcal{W}}$
World
$\mathcal{W}$
$\mathbf{H}^{\mathcal{WO}}(t)$
Object
$\mathcal{O}^{\mathcal{S}}(t)$
$\mathcal{O}^{\mathcal{W}}(t)$
$\mathbf{r}_1^{\mathcal{O}}$
$\mathbf{p}_m^{\mathcal{W}}$
$\mathbf{r}_n^{\mathcal{O}}$
Full problem

Figure 8.1: Identification of all structures and motions. Solutions $\mathbf{H}^{\mathcal{WS}}(t)$ and $\mathbf{H}^{\mathcal{OS}}(t)$ to separate problems for each sensor-object pair are merged by imposing a unique sensor motion (solid arrow). Assigning one object to be the world, the remaining moving objects can be specified in this world frame (object localization) with $\mathbf{H}^{\mathcal{WO}} = \mathbf{H}^{\mathcal{WS}}(\mathbf{H}^{\mathcal{OS}})^{-1}$; or in sensor frame (object tracking) with $\mathbf{H}^{\mathcal{SO}} = (\mathbf{H}^{\mathcal{OS}})^{-1}$. Boxes are frames; circles are points (world landmarks or object points).

3. An *incremental formulation* of these kind of solutions is *not evident*. This leads to methods that have to reconstruct the whole 3D scene at every frame, almost from scratch, or at least that have to take into account the whole set of points of every object, thus increasing the computational load as the complexity of the scene increases. This situation could be naturally relaxed by using filtering solutions because of their predicting capabilities.

We judge these impediments too severe to undertake a real-time executable solution to the problem with bearings-only sensors. In any case, such solutions would require the development of algorithms which are far apart from the SLAM line we have traced so far. Our position at this point is that MOT should be undertaken with range-and-bearing sensing.

8.3.2 Range-and-bearing sensing

In the previous chapter we developed BiCamSLAM, a method to acquire full 3D observability of nearby objects, while keeping the advantages that observation of remote landmarks provide, with the robustness of a continuously self-calibrated mechanical structure. The fundamental drawback of a limited-range 3D sensing is the impossibility to consider moving objects beyond the 3D observability region bounds.[1] Therefore, we limit our scope to those objects situated within these bounds. With this assumption in mind, we highlight the following key advantages of 3D sensing:

[1] This drawback is not exclusive to stereo or bi-camera systems: in mono-camera, the same considerations apply via numerical conditioning (disadvantage 2 above).

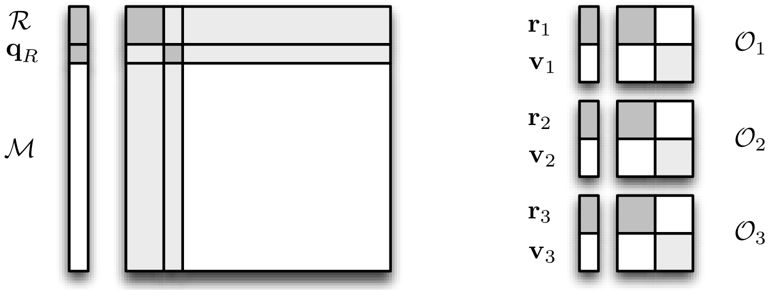

Figure 8.2: The SLAM-MOT map consists of the main BiCamSLAM map and a set of EKF, one per moving object.

1. *Trajectory observability* is out-of-doubt: if all successive positions are 3D observable, velocities are too.

2. *No need for previous segmentation*: as we can observe individual points, segmentation can be performed *a-posteriori*, based for instance on clustering of velocities and positions in the 3D space.

3. Straightforward *incremental formulation*: a dynamic model can be associated to each moving 3D point, which will be updated with filtering techniques. This model can easily include velocity information, which will allow us to make better predictions and also to feed the segmentation procedures above.

8.4 The Filtering side: Moving Objects Tracking

As indicated in Chapter 4, a SLAM system should use stable landmarks in order to guarantee proper localization. In the presence of moving objects, which are not stable by definition, two questions arise: how to determine what is moving and what is static; and what to do with moving points —how to take them into account.

On a first stage, we will consider the first question to be solved: we suppose that we know which points are mobile and which ones are static. With the static ones we perform usual SLAM. In this section we consider how to deal with the moving ones.

8.4.1 Uncorrelated moving objects

The inherent non-stability of moving objects avoids us to consider them as references for localization: the measurements the robot will make on them must not contribute to modify its belief on its own localization in any sense. This leads to an objects representation completely independent from the map, which in the EKF case is accomplished by de-correlating objects states from robot and landmarks states (Fig. 8.2). The only links between the robot pose, the mapped objects and their past positions are described by the current measurements and the object's motion models, which for the sake of simplicity are initially considered constant-velocity models.

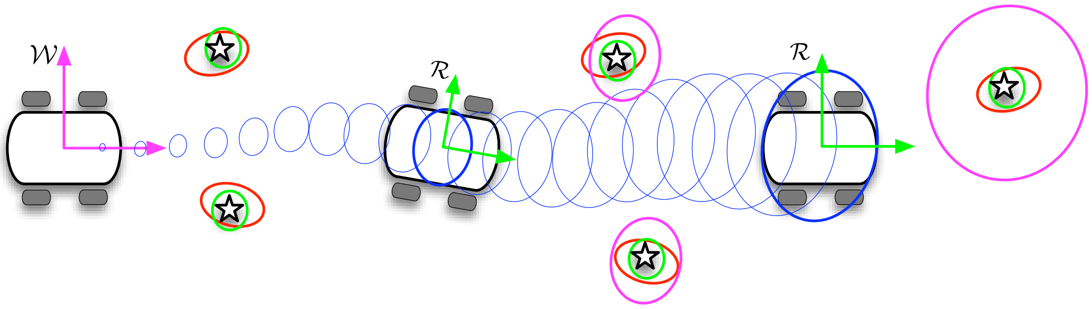

Figure 8.3: Objects global localization (*magenta*) versus local tracking (*green*). World and robot frames are shown in the corresponding colors. Robot estimates are in *blue*. Observations in robot frame are in *red*.

8.4.2 Global localization or local tracking?

The above de-correlation has a significant impact on the representation we must adopt for the objects states. We differentiate these representations as follows (Fig. 8.3):

- *Global localization.* We understand by this term the fact of having the objects positions and velocities expressed in the same frame as the map: the world frame. Object observations are composed with robot pose to result in object localization. This operation also composes overall uncertainties which add-up consequently: observe how objects uncertainties increase as robot uncertainty increases. As robot-objects cross-correlations are not maintained, this representation cannot preserve the accuracy of the original robot-objects relations established by the observations.

- *Local tracking.* In this case object states are expressed with respect to the current robot frame. By doing this, the cross-correlations between objects positions and the rest of the map (robot and landmarks) are null and there is no need to store them. The information provided by the observations is adequately preserved in the relative robot-objects positions.

You may at this point meditate about the famous sentence "*Think globally; act locally.*" For me, this is simple: with respect to the world, it is unclear where I am; with respect to my closest things, I have no doubt: I am '*here*' and they are '*right there*'. You may also consider this one: "*For your acts you'll be judged!*", which when combined with the previous one resumes to "*Whatever you think, it's your local action that matters*". I am not sure of where all this leads —just to say that we need a local representation for moving objects.

8.4.3 System set-up

We describe the elements to build a system to perform SLAM with Moving Objects Tracking based on filtering techniques and visual perception. This system is set-up as follows (Fig. 8.4):

1. The robot $\mathcal{R}^{\mathcal{W}}(t)$ is specified by its position and orientation in the 3D space or world frame $\mathcal{W}$;

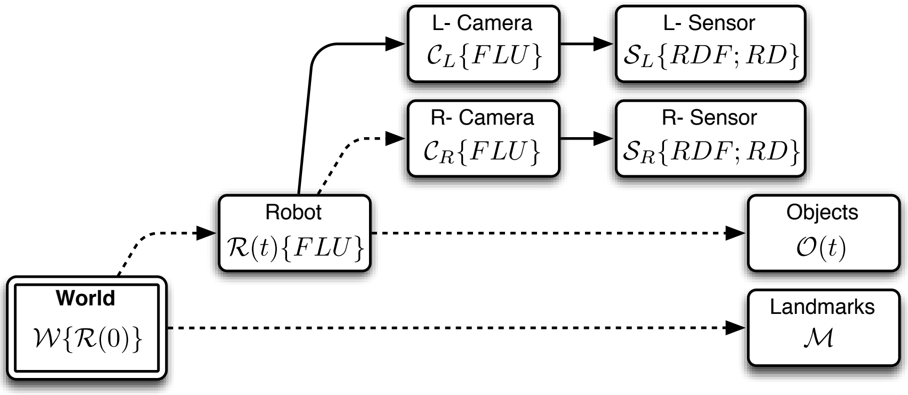

Figure 8.4: Reference frames diagram used in our SLAM-MOT solution (refer to Chapter 1 for notation). Solid-line arrows indicate deterministic definitions. Dashed-line ones are uncertain; their *pdfs* are continuously estimated by BiCamSLAM-MOT.

2. The mobile objects $\mathcal{O}_i^{\mathcal{R}}(t)$ are considered punctual, defined by their position and linear velocities in the robot frame;

3. The landmarks $\mathbf{p}$ are as usual related to world frame; all that relates to landmarks receives the usual SLAM treatment and is omitted in this presentation

4. The notation indicating the reference frames, the notion of time (t) and the object indices $(\cdot)_i$ are considered implicit and omitted in the formulation.

We write then the representations of robot and mobile objects states as follows:

$$\mathcal{R} = \begin{bmatrix} \mathbf{x} \\ \mathbf{q} \end{bmatrix} \qquad \mathcal{O} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix}. \tag{8.1}$$

It will be also convenient to clarify some terminology. We give the following definitions:

Definition 8.1 (Point). A generic 3D point in space with no particularly specified role. □

Definition 8.2 (Landmark). A steady point which is intended for mapping and robot localization. □

Definition 8.3 (Object). A point (normally but not necessarily moving) which is intended to be tracked by the robot. Sometimes we will also use the terms 'moving object' or 'mobile', or other fancy combinations. □

Definition 8.4 (Solid). A set of nearby points with coherent motion (very similar velocities). □

Definition 8.5 (World). A special solid defined by the set of all landmarks. □

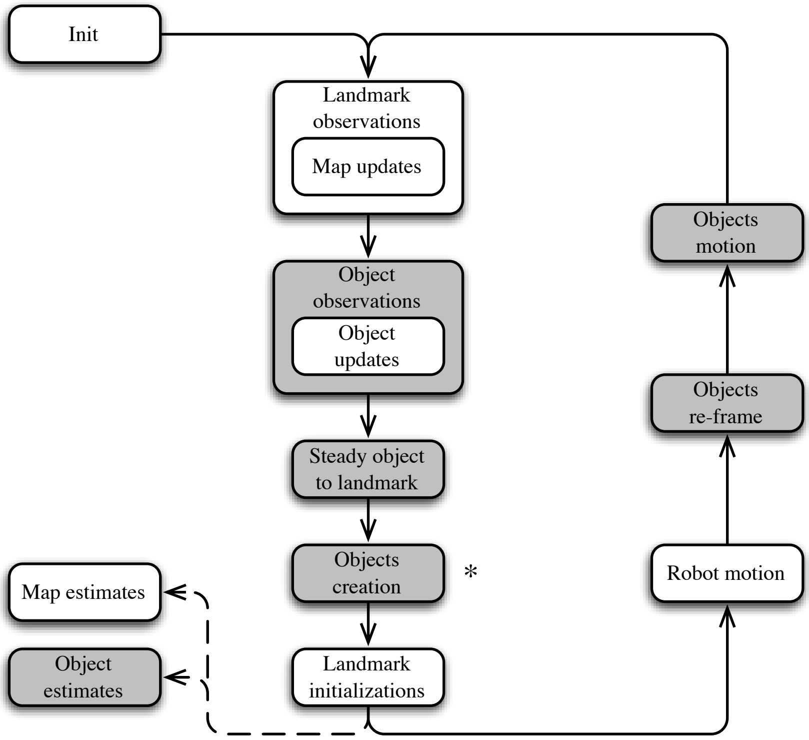

Figure 8.5: The SLAM-MOT algorithm. Gray boxes correspond to operations specific to Moving Objects Tracking.

8.4.4 The algorithm

Mathematically speaking, there is nothing new to add. As we revise now, we have all the material to set up such a SLAM system with moving objects tracking. The BiCamSLAM algorithm is modified to accommodate MOT as indicated in Fig. 8.5. Starting at the original creation of an object (marked by an asterisk in the figure) and proceeding in the loop direction, the different operations are illustrated in Fig. 8.6 and described in the pages that follow.

MOT: Objects creation

Upon each mobile object detection –a complex problematic that we will tackle later– an EKF is created to host its state's *pdf*. The mobile state vector is hence the Gaussian $\mathcal{O} \sim \mathcal{N}\{\bar{\mathcal{O}}; \mathbf{P}_{\mathcal{O}}\}$. For its creation we will follow the BiCam initialization methods because full observability is needed: those points beyond the full-observability region are discarded as mobile object candidates.

Initial mean and covariances matrix for the object's position $\mathbf{r}$ are determined from the inverse observation function of the left-hand camera

$$\mathbf{r} = \mathbf{g}_L(\mathcal{C}_L, \mathbf{b}_L, s)$$

where $\mathcal{C}_L^\top = [\mathbf{x}_L^\top, \mathbf{q}_L^\top]$ is the left-hand camera frame which is absolutely deterministic, $\mathbf{b}_L \sim \mathcal{N}\{\mathbf{y}_L; \mathbf{R}\}$ is the Gaussian observation and $s \sim \mathcal{N}\{\bar{s}; \sigma^2\}$ is the Gaussian depth whose parameters are determined with the BiCam methods. We remind the implementation of $\mathbf{g}_L(\cdot)$ in

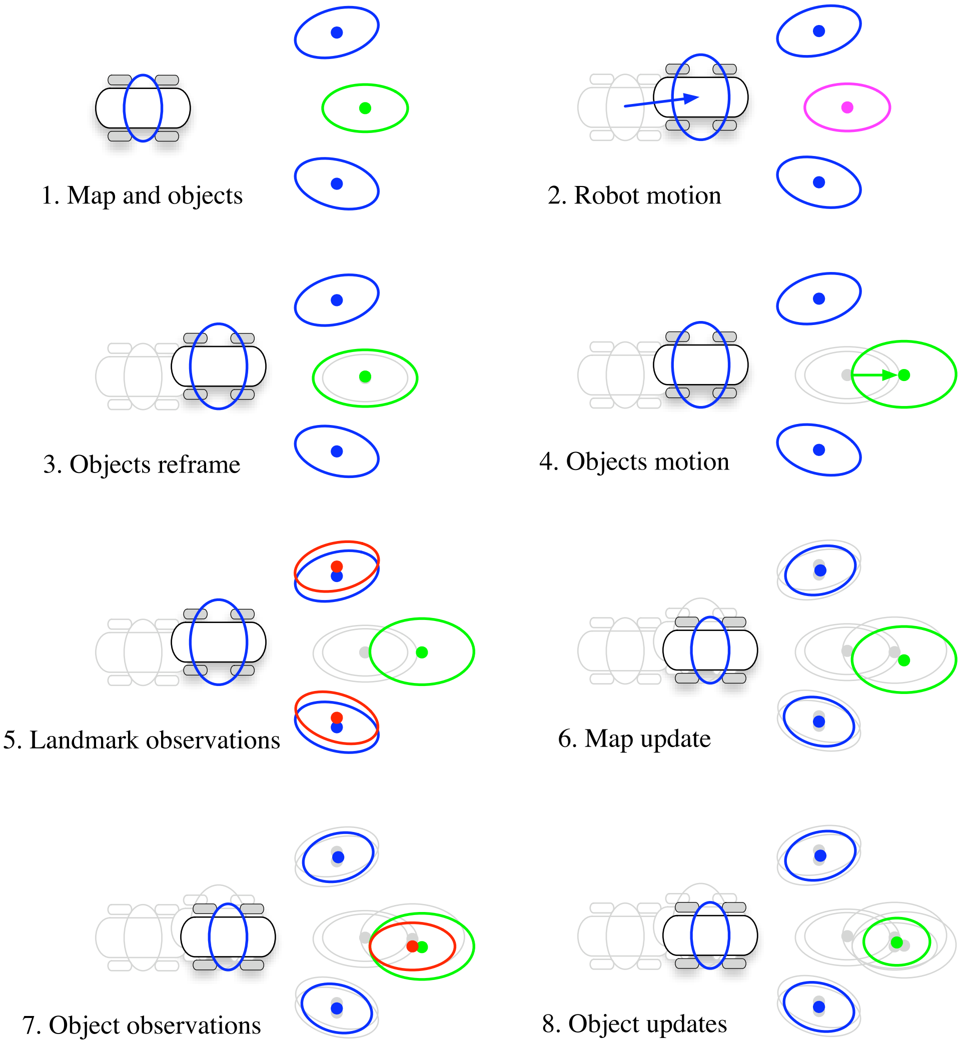

Figure 8.6: A complete filter loop in SLAM-MOT. Map estimates are in *blue*. Objects in the current robot frame in *green*. Objects in the old robot frame in *magenta*. Observations in *red*. 1) Initial robot, landmarks and objects estimates. 2) Robot motion increases robot uncertainty. 3) Objects reframe translates this extra uncertainty to the objects. 4) Upon motion, objects further increase their uncertainty. 5,6) Landmarks observations are used to update the map. See that object estimates move with the robot frame as this frame is updated. 7,8) Object observations and updates refine the object's motion model. They have no effect on the map.

function form[2] as given in Chapter 2:

$$\begin{aligned} \mathbf{r}^{\mathcal{S}_L} &= backProject\Big(s, correct\big(\mathbf{c}_L, unpixellize(\mathbf{k}_L, \mathbf{b}_L)\big)\Big) \\ \mathbf{r} &= fromFrame(\mathcal{C}_L, \mathbf{R}^{\mathcal{CS}}\,\mathbf{r}^{\mathcal{S}_L}). \end{aligned}$$

where $\mathbf{k}_L$ and $\mathbf{c}_L$ are the left-hand's camera intrinsic and distortion correction parameters and $\mathbf{R}^{\mathcal{CS}}$ is the camera-to-sensor rotation matrix. Object position's initial mean and covariances matrix are then

$$\bar{\mathbf{r}} = \mathbf{g}_L(\mathcal{C}_L, \bar{\mathbf{b}}_L, \bar{s}) \tag{8.2}$$

$$\mathbf{P}_{\mathbf{rr}} = \mathbf{G}_{L\mathbf{b}}\,\mathbf{R}\,\mathbf{G}_{L\mathbf{b}}^\top + \mathbf{G}_{Ls}\,\sigma^2\,\mathbf{G}_{Ls}^\top \tag{8.3}$$

where the Jacobian matrices are[3]

$$\mathbf{G}_{L\mathbf{b}} = \left.\frac{\partial \mathbf{g}_L}{\partial \mathbf{b}^\top}\right|_{(\bar{\mathcal{C}}_L, \bar{\mathbf{b}}, \bar{s})} \qquad \mathbf{G}_{Ls} = \left.\frac{\partial \mathbf{g}_L}{\partial s^\top}\right|_{(\bar{\mathcal{C}}_L, \bar{\mathbf{b}}, \bar{s})}$$

Initial mean and covariances matrix for the object's velocity $\mathbf{v} \sim \mathcal{N}\{\bar{\mathbf{v}}; \mathbf{P}_{\mathbf{vv}}\}$ are heuristically determined. With scenario-dependent previous knowledge (about the scene characteristics and the important objects to be detected) velocity initializations can be differently defined depending on the region in the image where they have been detected. This will be further developed in Section 8.5.

The full object's Gaussian *pdf* is then specified by the couple[4]

$$\bar{\mathcal{O}} = \begin{bmatrix} \bar{\mathbf{r}} \\ \bar{\mathbf{v}} \end{bmatrix} \qquad \mathbf{P}_{\mathcal{O}} = \begin{bmatrix} \mathbf{P}_{\mathbf{rr}} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\mathbf{vv}} \end{bmatrix}. \tag{8.4}$$

Immediately after creation, this EKF is updated with the observation from the right-hand camera (described a few paragraphs below).

MOT: Robot motion

The robot's time-evolution model is generically written as follows:

$$\mathcal{R}^+ = \mathbf{f}_{\mathcal{R}}(\mathcal{R}, \mathbf{u}).$$

It responds to the odometry model introduced in Chapter 1 as

$$\begin{aligned} \mathbf{x}^+ &= fromFrame(\mathcal{R}, \delta\mathbf{x}) \\ \mathbf{q}^+ &= \mathbf{q} + \frac{1}{2}\,\boldsymbol{\Omega}(\delta\mathbf{e})\,\mathbf{q} \end{aligned}$$

[2] Although homogeneous representations are cleaner, we use function forms because they better relate to the Jacobian matrices that we will have to define. See Appendix A for clues on obtaining the Jacobian matrices of composed functions.

[3] We give the Jacobian matrices for this case an omit their definitions for the rest of the chapter. At this point of the document this should not trouble the reader.

[4] In the case where these initial velocities also depend on the determined location in the 3D space, we could additionally have cross-correlations between the position and velocity components. At this point of the development this would just complicate things up.

where $\mathbf{u}$ is the vector of robot controls or odometry data

$$\mathbf{u} = \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{e} \end{bmatrix} = [\delta x, \delta y, \delta z, \delta\phi, \delta\theta, \delta\psi]^\top \in \mathbb{R}^6$$

with Gaussian *pdf* $\mathbf{u} \sim \mathcal{N}\{\bar{\mathbf{u}}; \mathbf{U}\}$. The skew-symmetric matrix $\mathbf{\Omega}(\delta\mathbf{e})$ was also defined to be

$$\mathbf{\Omega}(\delta\mathbf{e}) = \begin{bmatrix} 0 & -\delta\phi & -\delta\theta & -\delta\psi \\ \delta\phi & 0 & \delta\psi & -\delta\theta \\ \delta\theta & -\delta\psi & 0 & \delta\phi \\ \delta\psi & \delta\theta & -\delta\phi & 0 \end{bmatrix}.$$

Except for the change in the reference frame which is described in the next paragraph, robot motion has no effect on mobile objects. The prediction operation to be performed on the SLAM map is as usual and not described here.

MOT: Objects re-framing

Upon robot motion, mobile objects must change their coordinates from the old robot frame to the new one. This re-framing operation will be generically written as follows:[5]

$$\mathcal{O}^+ = \mathbf{j}(\mathcal{O}, \mathbf{u})$$

which is a frame transformation $\mathcal{R} \to \mathcal{R}^+$ that is specified by the odometry parameters $\mathbf{u}^\top = [\delta\mathbf{x}^\top, \delta\mathbf{e}^\top]$. It just requires the computation of a rotation matrix $\mathbf{R}(\delta\mathbf{e})$ from the Euler angles increments $\delta\mathbf{e}$ (Appendix C). If we name $(\mathcal{R}^+)^\mathcal{R}$ the new robot frame with respect to the old one, we can then write the mobile object in the new robot frame as:

$$\begin{aligned} \mathbf{r}^+ &= toFrame((\mathcal{R}^+)^\mathcal{R}, \mathbf{r}) &= \mathbf{R}^\top(\delta\mathbf{e}) \cdot (\mathbf{r} - \delta\mathbf{x}) \\ \mathbf{v}^+ &= \overline{toFrame}((\mathcal{R}^+)^\mathcal{R}, \mathbf{v}) &= \mathbf{R}^\top(\delta\mathbf{e}) \cdot \mathbf{v} \end{aligned}$$

where $\overline{toFrame}()$ means the function applied to a vector (recall Remark 1.1). Observe that the new mobile positions and velocities are still disconnected from the global frame $\mathcal{W}$: the transformations are just driven by the robot motion parameters.

Recalling that $\mathbf{u} \sim \mathcal{N}\{\bar{\mathbf{u}}; \mathbf{U}\}$ the object's *pdf* is updated following the EKF equations as follows

$$\bar{\mathcal{O}}^+ = \mathbf{j}(\bar{\mathcal{O}}, \bar{\mathbf{u}}) \tag{8.5}$$

$$\mathbf{P}_\mathcal{O}^+ = \mathbf{J}_\mathcal{O} \, \mathbf{P}_\mathcal{O} \, \mathbf{J}_\mathcal{O}^\top + \mathbf{J}_\mathbf{u} \, \mathbf{U} \, \mathbf{J}_\mathbf{u}^\top \tag{8.6}$$

where we will notice the object's uncertainty increase due to the term $\mathbf{J}_\mathbf{u} \, \mathbf{U} \, \mathbf{J}_\mathbf{u}^\top$.

MOT: Objects motion

The mobile object's time-evolution model is generically written as:

$$\mathcal{O}^+ = \mathbf{f}_\mathcal{O}(\mathcal{O}, \omega).$$

[5] Notice that we use the following letters to indicate generic functions: $\mathbf{f}$ for time-evolution; $\mathbf{g}$ for inverse-measurements; $\mathbf{h}$ for measurements; and now $\mathbf{j}$ for static re-framing. This is handy in order to name their Jacobian matrices via $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$ and $\mathbf{J}$ as we do.

It responds to the constant-velocity model introduced in Chapter 1, with no rotational part, which is detailed as follows:

$$\mathbf{r}^+ = \mathbf{r} + T_s\,\mathbf{v} \tag{8.7}$$

$$\mathbf{v}^+ = \mathbf{v} + \omega \tag{8.8}$$

where T_s is the filter's sampling time and $\omega = [\omega_x, \omega_y, \omega_z]^\top \in \mathbb{R}^3$ is a white Gaussian velocity perturbation $\omega \sim \mathcal{N}\{0; \mathbf{Q}\}$.

The object's *pdf* is updated as follows

$$\bar{\mathcal{O}}^+ = \mathbf{f}_{\mathcal{O}}(\bar{\mathcal{O}}, 0) \tag{8.9}$$

$$\mathbf{P}_{\mathcal{O}}^+ = \mathbf{F}_{\mathcal{O}\mathcal{O}}\,\mathbf{P}_{\mathcal{O}}\,\mathbf{F}_{\mathcal{O}\mathcal{O}}^\top + \mathbf{F}_{\mathcal{O}\omega}\,\mathbf{Q}\,\mathbf{F}_{\mathcal{O}\omega}^\top. \tag{8.10}$$

Notice the uncertainty increase due to the term $\mathbf{F}_{\mathcal{O}\omega}\,\mathbf{Q}\,\mathbf{F}_{\mathcal{O}\omega}^\top$.

MOT: Objects observations

The objects observation functions are now defined in the robot frame. As we dispose of two cameras, these functions are generically written as

$$\begin{aligned}\mathbf{y}_L &= \mathbf{h}_L(\mathcal{C}_L, \mathbf{r}) + \upsilon_L\\ \mathbf{y}_R &= \mathbf{h}_L(\mathcal{C}_R, \mathbf{r}) + \upsilon_R\end{aligned}$$

where $\{\upsilon_L, \upsilon_R\} \sim \mathcal{N}\{0; \mathbf{R}\}$ are independent white Gaussian observation noises which, for the sake of simplicity, are supposed to have the same variance —though this is not a requirement. This functions are defined as in Chapter 2. We give that of the right-hand camera as a reminding example:

$$\begin{aligned}\mathbf{r}^{\mathcal{S}_R} &= \mathbf{R}^{\mathcal{S}\mathcal{C}} \cdot toFrame(\mathcal{C}_R, \mathbf{r})\\ \mathbf{h}_R(\mathcal{C}_R, \mathbf{r}) &= pixellize\Big(\mathbf{k}_R, distort\big(\mathbf{d}_R, project(\mathbf{r}^{\mathcal{S}_R})\big)\Big)\end{aligned}$$

where $\mathcal{C}_R^\top = [\mathbf{x}_R^\top, \mathbf{q}_R^\top]$ is the right-hand camera frame in the robot, $\mathbf{R}^{\mathcal{S}\mathcal{C}} = (\mathbf{R}^{\mathcal{C}\mathcal{S}})^\top$ is the sensor-to-camera rotation matrix, and $\mathbf{k}_R$ and $\mathbf{d}_R$ are the right-hand camera's intrinsic and distortion parameters.

Upon observation from this right-hand camera, which may have an uncertain pose $\mathcal{C}_R \sim \mathcal{N}\{\bar{\mathcal{C}}_R; \mathbf{P}_{\mathcal{C}_R}\}$, the object's *pdf* receives the following EKF update[6]

$$\mathbf{z}_R = \mathbf{y}_R - \mathbf{h}_R(\bar{\mathcal{C}}_R, \bar{\mathbf{r}}) \tag{8.11}$$

$$\mathbf{Z}_R = \mathbf{H}_{R\mathcal{O}}\,\mathbf{P}_{\mathcal{O}}\,\mathbf{H}_{R\mathcal{O}}^\top + \mathbf{H}_{R\mathcal{C}}\,\mathbf{P}_{\mathcal{C}_R}\,\mathbf{H}_{R\mathcal{C}}^\top + \mathbf{R} \tag{8.12}$$

$$\mathbf{K} = \mathbf{P}_{\mathcal{O}}\,\mathbf{H}_{R\mathcal{O}}^\top\,\mathbf{Z}_R^{-1} \tag{8.13}$$

$$\bar{\mathcal{O}}^+ = \bar{\mathcal{O}} + \mathbf{K}\,\mathbf{z}_R \tag{8.14}$$

$$\mathbf{P}_{\mathcal{O}}^+ = \mathbf{P}_{\mathcal{O}} - \mathbf{K}\,\mathbf{Z}_R\,\mathbf{K}^\top. \tag{8.15}$$

We recall for practical purposes that the innovation $\{\mathbf{z}_R; \mathbf{Z}_R\}$ is obtained from the expectation $\mathbf{e}_R \sim \mathcal{N}\{\bar{\mathbf{e}}_R; \mathbf{E}_R\}$ defined by

$$\begin{aligned}\bar{\mathbf{e}}_R &= \mathbf{h}_R(\bar{\mathcal{C}}_R, \bar{\mathbf{r}})\\ \mathbf{E}_R &= \mathbf{H}_{R\mathcal{O}}\,\mathbf{P}_{\mathcal{O}}\,\mathbf{H}_{R\mathcal{O}}^\top + \mathbf{H}_{R\mathcal{C}}\,\mathbf{P}_{\mathcal{C}_R}\,\mathbf{H}_{R\mathcal{C}}^\top + \mathbf{R}\end{aligned}$$

whose 3σ ellipse is previously used for matching via the active-search methods.

[6] Notice that for the left-hand camera we have $\mathbf{P}_{\mathcal{C}_L} = 0$.

MOT: Landmark initialization from a steady object

Very often the created mobile objects do not correspond to moving points but steady. After evaluating velocity means and covariances, if we conclude that this is effectively the case, this object should be transformed into a landmark and be included in the SLAM map via landmark full initialization (as in Chapter 4). This operation consists in a simple reference transformation from robot- to world- frame:

$$\mathbf{p} = fromFrame(\mathcal{R}, \mathbf{r}) = \mathbf{R}(\mathbf{q}) \cdot \mathbf{r} + \mathbf{x}.$$

The SLAM map receives a full landmark initialization as follows:

$$\bar{X}^+ = \begin{bmatrix} \bar{X} \\ \bar{\mathbf{p}} \end{bmatrix} \tag{8.16}$$

$$\mathbf{P}^+ = \begin{bmatrix} \mathbf{P} & \mathbf{P}_{pX}^\top \\ \mathbf{P}_{pX} & \mathbf{P}_{pp} \end{bmatrix} \tag{8.17}$$

with

$$\bar{\mathbf{p}} = \mathbf{R}(\bar{\mathbf{q}}) \cdot \bar{\mathbf{r}} + \bar{\mathbf{x}} \tag{8.18}$$

$$\mathbf{P}_{pX} = \mathbf{FF}_\mathcal{R}\, \mathbf{P}_{\mathcal{R}X} \tag{8.19}$$

$$\mathbf{P}_{pp} = \mathbf{FF}_\mathcal{R}\, \mathbf{P}_{\mathcal{R}\mathcal{R}}\, \mathbf{FF}_\mathcal{R}^\top + \mathbf{R}(\bar{\mathbf{q}})\, \mathbf{P}_{\mathbf{rr}}\, \mathbf{R}^\top(\bar{\mathbf{q}}) \tag{8.20}$$

where $\mathbf{FF}_\mathcal{R} = \left.\frac{\partial fromFrame(\mathcal{R},\mathbf{r})}{\partial \mathcal{R}^\top}\right|_{\bar{\mathcal{R}},\bar{\mathbf{r}}}$ is the Jacobian matrix of $fromFrame(\cdot)$ with respect to the robot frame as defined in Appendix A.

MOT: Object estimates in world frame

Just for estimation and visualization purposes, mobile objects must sometimes be referred to the world frame. This is accomplished simply with

$$\begin{aligned} \hat{\mathbf{r}}^\mathcal{W} &= fromFrame(\bar{\mathcal{R}}, \bar{\mathbf{r}}) &= \mathbf{R}(\bar{\mathbf{q}}) \cdot \bar{\mathbf{r}} + \bar{\mathbf{x}} \\ \hat{\mathbf{v}}^\mathcal{W} &= \overline{fromFrame}(\bar{\mathcal{R}}, \bar{\mathbf{v}}) &= \mathbf{R}(\bar{\mathbf{q}}) \cdot \bar{\mathbf{v}}. \end{aligned}$$

where $\mathbf{R}(\mathbf{q})$ is the rotation matrix corresponding to the robot orientation (defined in Appendix C) and $\mathbf{x}$ is the robot position. The position covariances matrix $\mathbf{P}_{\mathbf{rr}}$ often needs to be referred to the world frame for visualization (*i.e.* to draw its 3σ ellipsoids). It can be obtained with

$$\mathbf{P}_{\mathbf{rr}}^\mathcal{W} = \mathbf{R}(\bar{\mathbf{q}})\, \mathbf{P}_{\mathbf{rr}}\, \mathbf{R}^\top(\bar{\mathbf{q}}) \tag{8.21}$$

where the robot uncertainty is considered null as we dispose of no cross-correlation information.

8.5 The Perception side: Moving Objects Detection

The perception side of SLAM-MOT is a delicate task. Moving objects may be relatively easy to track,[7] but they are by no means easy to detect: they can appear at any time, in any

[7] Just project the estimates into the image, draw the 3σ ellipsoids of their expectations and perform active search. A double EKF-update, one per camera, completes the tracking loop.

region of the image; they can come out from behind an obstacle; they can suddenly stop, start or change their direction; and the most important: they should not be missed by our vision system. With such a sparse punctual representation of reality as the one we use and with the limited computer resources that we have, this is not easy work.

As we did with active features search, we try to find procedures to guide the detection algorithms so that we maximize their chances to success while economizing computer resources. We do this by exploiting the knowledge we have on the system, and by developing the following ideas:

Under certain circumstances that the system will anticipate, a new detected feature may be considered as belonging to a moving object. This object will be assigned a constant velocity model with a particular initial estimate based on some *a-priori* knowledge. If we are right and the perceived object is effectively moving, we got it.[8] If we are wrong, its velocity will converge to zero and, this case being detectable, it can be later included as part of the static world. In the other direction, detected points that are initially supposed static (thus belonging to the world) may eventually belong to moving objects. In this case the system will rapidly loose them and, this situation being also detectable, we can immediately search for a moving thing in the vicinity.

We found that different strategies can be imagined to fulfill these objectives. They correspond to different situations where moving objects could arise, which can be anticipated by the system. We call these strategies the *Detection Mechanisms* (DM). Unfortunately, each DM needs to be executed in a different place in the filter loop sequence and a unified algorithm becomes almost impossible. In order to give the necessary elasticity to the system in front of this variety of strategies, we define a two-step algorithm: first, each DM is run independently and gives its intentions to search a particular area in the image (we may see this as a voting strategy). Second, a *Priority-based Detection Selector* (PbDS) is used to select which regions will be effectively searched, what kind of point we will search (moving object or landmark), and with which initial conditions (velocity magnitudes, directions and uncertainties). These two steps are explained in the following sections.

8.5.1 Detection Mechanisms (DM)

Object detection is performed at image level (2D reasoning) based on three[9] different *detection mechanisms*:

1. *Intentional detection.* Certain image regions are considered crucial for moving objects detection. Detected points in these regions are systematically assigned to mobile objects.

2. *Event-driven detection.* Detectable events such as occlusions can indicate the presence of unexpected objects. Moving objects are systematically searched around those features that suddenly failed to match.

3. *Object growing detection.* New moving points will be searched in the vicinity of those that already move. These points will belong, with high probability, to the same moving solid.

[8] Of course we assume that the tracker is working.

[9] They could be more than three: as they act independently, every new source of evidence for a moving object can be incorporated as a DM.

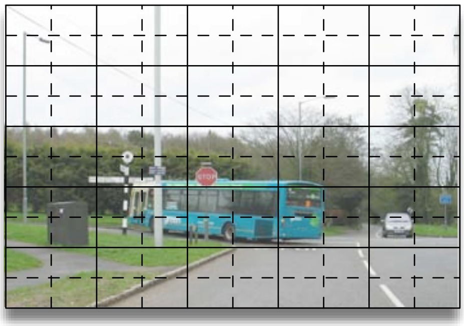

Figure 8.7: Two differently spaced grids (*solid* and *dashed* lines) define the cells where landmarks and moving objects will be searched. The different detection mechanisms will simply activate those cells with sufficient interest to be searched.

In order to unify these procedures to coexist with the necessary landmarks detector, we propose the following strategy:

Take the left image as the reference. This image is cut into two different grids of cells as shown in Fig. 8.7. A thicker grid of $N \times M$ cells is used for landmarks detection as in the previous chapters (we show a 5×5 cells grid). In order to provide a higher density of moving objects, a thinner grid of $n \times m$ cells, where $\{n, m\}$ are integer multiples of $\{N, M\}$, is used for objects detection (of double density for instance, *i.e.* $\{n, m\} = \{2N, 2M\}$, 10×10 cells in the figures). The cells of these grids are the basic units for feature detection, *i.e.* those that will be analyzed by the corner detectors presented in Chapter 2.

An empty sample of the thicker grid is assigned to the landmarks DM in the form of a binary matrix. Name this matrix the '*Landmarks binary matrix*' S_L, which is size $N \times M$. Similarly, each one of the objects DM will receive an empty copy of the thin grid, also in the form of binary matrices of size $n \times m$. Name these matrices S_I, S_E and S_G for '*Intentional-*', '*Event-driven-*' and '*object-Growing-*' *binary matrices*. They are also given structure '*velocity matrices*' V_I, V_E and V_G to host the *a-priori* velocity estimates (*i.e.* mean and covariances matrix of $\mathbf{v} \sim \mathcal{N}\{\bar{\mathbf{v}}; \mathbf{P_{vv}}\}$). These constructions are summarized in Table 8.2.

Table 8.2: Matrices holding cells information generated by each Detection Mechanism.

DM	Grid size	Binary matrix	Velocity matrix
Landmarks	$N \times M$	S_L	
Intentional	$n \times m$	S_I	V_I
Event-driven	$n \times m$	S_E	V_E
Object growing	$n \times m$	S_G	V_G

With the aid of these matrices, each one of the landmark- and object- DM (which will be individually detailed soon) will perform the following tasks on their respective grids:

- Activate those cells where a new feature should be searched. For cell (i, j), this is simply done by setting the respective binary matrix entry:

$$S_X(i, j) = 1.$$

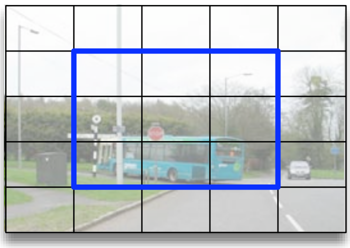

Figure 8.8: Landmarks DM active cells.

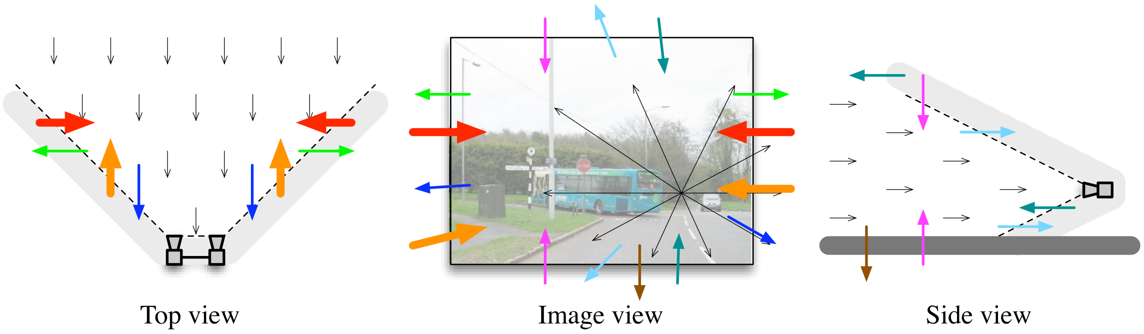

Figure 8.9: Intentional moving objects detection. Top, side and image views are shown with apparent motions with respect to the robot frame. Landmarks motions are in thin black arrows. The only mobiles that are interesting to be tracked are those entering the image by the sides (*red* and *orange* trajectories).

with $X \in \{I, E, G, L\}$.

- Only for object's active cells, define initial velocity estimates for the objects to create in case of successful detection. This is accomplished by setting the velocity matrix:

$$\begin{aligned} V_X(i,j).v &= \bar{\mathbf{v}} \\ V_X(i,j).P &= \mathbf{P_{vv}} \end{aligned}$$

with $X \in \{I, E, G\}$.

It follows a detailed description of the three objects DM, as-well as a small remainder of the landmarks DM.

DM: Landmarks detection (remainder)

Landmarks will be searched in the inner cells of the grid. These cells are systematically activated as indicated in Fig. 8.8.

DM: Intentional objects detection

Intentional mobiles detection tries to detect moving objects entering the robot's field of view. This concerns the cells adjacent to image edges. Possible trajectories of objects in these cells are illustrated in Fig. 8.9. Only the objects effectively *entering* the image are considered

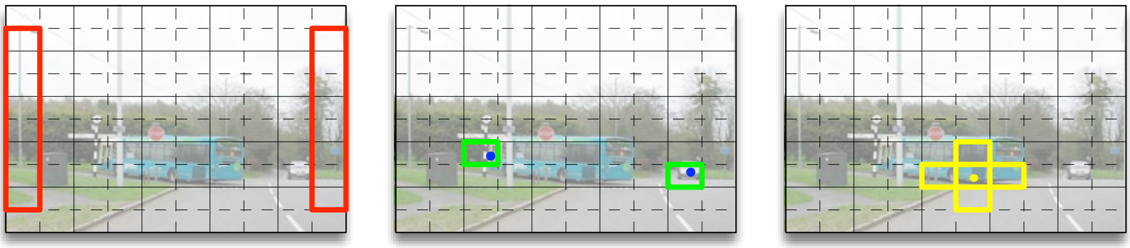

Figure 8.10: Active cells for moving objects DM. Intentional detection activates a fixed cells distribution (*left*). Event-driven detection activates cells with features that failed to match (*center*; failed matches as *blue* dots). Object-growing detection activates neighbor cells to that of an already existing moving object (*right*; existing object as *yellow* dot).

because, besides being the only ones that can be tracked (the other ones will soon abandon the image), they are the only ones to have trajectories interfering with the robot's. Among them, those entering by the upper and lower edges represent highly improbable objects (no falling pianos are expected!) and are not considered.

Thus only the mobiles marked *red* and *orange* in the figure should be considered. This allows us to:

- Define a fixed structure of active cells like that shown in Fig. 8.10 (*left*).
- Define their *a-priori* velocity estimates in 3D space $\mathbf{v} \sim \mathcal{N}\{\bar{\mathbf{v}}; \mathbf{P_{vv}}\}$, differently for left- and right-hand side entering objects.

DM: Event-driven objects detection

By this mechanism we try to detect objects in the central regions of the image. In order to avoid a random search, we choose this event-driven approach which will focus the attention where interesting things are happening.

This method is executed at the time of feature observation: a failure on the measurement validation test[10] (Section 6.5.3 in Chapter 6) will be interpreted as an eventual moving object occluding the expected 3D point and will consequently fire the following two actions:

- The cell containing the missed feature is activated for later object detection as in Fig. 8.10 (*center*).
- The velocity *a-priori* is set to zero-mean speed and a sufficiently large velocity covariances matrix in order to 'catch' the eventual object.

DM: Objects growing detection

This mechanism tries to detect moving objects (points) belonging to the same real solid as an existing one. It does this by:

- Activating neighbor cells as in Fig. 8.10 (*right*).
- Defining the velocity's *a-priori pdf* equal to that of the original object.

[10] This test can be completed with a closer analysis of the ZNCC evolution during the last frames to better detect sudden mismatches: those which exhibit stable good matches with a sudden important fall of the ZNCC score.

8.5.2 Priority-based Detection Selector (PbDS)

Once all DM matrices are updated, a priority-driven fusion mechanism is executed to assign a unique action to each grid cell in the image: either do nothing; or search for a landmark; or search for a moving object.

When collecting all DM results together, we may encounter the following situations for each cell in the image:

1. The cell may contain features of already initialized landmarks or objects.

2. The cell may not be activated by any DM. In this case we do nothing.

3. The cell may be activated by the landmarks DM and some object DM.

4. The cell may be activated by more than one object DM.

5. The total number of active cells may be too large for the computer to perform all searches while keeping real-time operation.

We need some clear criterion to decide what to do in each individual cell. Besides computer resources considerations, we propose these three criterions related to the problem itself:

Criterion 8.1 (Priority by distinguishability):
Different features should be associated to different 3D points. ◇

Criterion 8.2 (Priority by motion):
Objects in motion should not be missed. ◇

Criterion 8.3 (Priority by certainty):
In case of doubt, the most informative hypothesis should be considered. ◇

Criterion 8.1 suggests that we should not search a new feature in those cells that already contain one.[11] There is one exception: event-driven detection is precisely defined in cells containing one feature: the missed one.

Criterion 8.2 suggests that we should try to detect moving objects before attempting with static landmarks. Depending on how we look at it, this may seem a little weird; but we already gave the means to transform a moving object into a static mapped landmark in case this object does not effectively move.

Criterion 8.3 suggests that, in case a cell is activated by more than one object detection mechanism, the most accurate velocity estimates should be assigned to the created object in case of successful detection.

We have almost all the material to present the PbDS fusion algorithm. Before that, we need to introduce some additional matrix constructions (summarized in Table 8.3 together with the previous ones):

- For the objects detector, a binary $n \times m$ matrix F_O with active entries for those cells in the image containing at least one feature (landmark or object).

[11] The Harris detector is blind and, by running it inside a cell containing a feature, it will probably re-detect this same feature.

Table 8.3: Matrices intervening in the priority-based detection selector.

Search type	Objects	Landmarks
Grid size	$n \times m$	$N \times M$
Existing features matrix	F_O	F_L
DM binary matrices	S_I, S_E, S_G	S_L
DM velocity matrices	V_I, V_E, V_G	
Output binary matrix	D_O	D_L
Output velocity matrix	V_O	

- For the landmarks detector, a binary $N \times M$ matrix F_L with active entries for those cells in the image containing at least one feature (landmark or object).
- An empty $n \times m$ binary matrix D_O that will be the result of the objects selector algorithm, *i.e.* the set of cells to be searched for objects.
- The corresponding resulting velocities matrix V_O, whose entries are the velocity *a-prioris* that a successfully detected object will receive.
- An empty $N \times M$ binary matrix D_L that will be the result of the landmarks selector algorithm, *i.e.* the set of cells to be searched for landmarks.

On these matrices, we define the following syntax:

- Boolean operations on binary matrices: And: $A \cap B$. Or: $A \cup B$. Not: $\overline{A}$.
- Multiple indexing: $Z(B)$ are those entries in structure matrix Z indexed by the corresponding *true* entries in the binary B. We may then write $T(B) = Z(B)$ which means that the entries in Z indexed by B are assigned to the corresponding entries in T, and that the rest of entries in T are unchanged.

PbDS: The algorithm

That's it. The Priority-based Detection Selector algorithm (illustrated in Figs. 8.11, 8.12 and 8.13.) can now be written as:

1. Project currently mapped landmarks and objects into the image. Count them in a $n \times m$ per-cell basis and set those cells in matrix F_O that contain at least one feature.
2. Similarly set the cells in matrix F_L by grouping the corresponding cells in F_O.
3. Landmarks DM has a fixed cell pattern. It provides the constant binary matrix S_L.
4. Get Objects DM matrices:
   (a) Intentional DM has a fixed cell pattern. It provides the constant matrices S_I and V_I.
   (b) Event-driven DM was performed at observation time giving S_E and V_E.

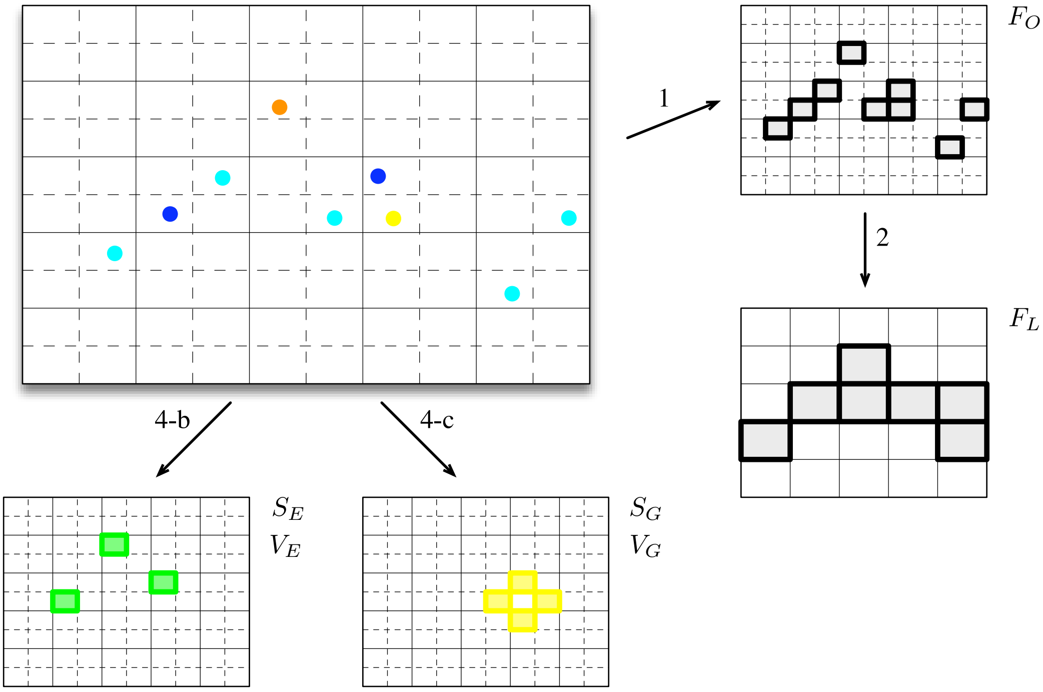

Figure 8.11: Priority-based Detection Selector algorithm -I-: Generation of matrices F_O, F_L, $\{S_E, V_E\}$ and $\{S_G, V_G\}$ from the set of projected 3D points in the image. Missed landmarks (*dark blue*) and missed objects (*orange*) activate the Event-driven DM cells (*green*); Matched objects (*yellow*) activate their four Object-growing DM neighbor cells. Matched landmarks are in *pale blue*. Existing features set cells in F_O which in turn set those in F_L (both in *gray*). Arrow numbers refer to the corresponding step in the algorithm.

(c) Perform Objects growing DM. Obtain S_G and V_G.

5. Assign the less informative velocity estimates (event-driven ones) to the objects' output velocity matrix:
$$V_O = V_E.$$

6. Set objects' output binary matrix cells with intentional DM cells and assign more informative velocity estimates (intentional ones; overwrite eventual existing velocities):
$$\begin{aligned} D_O &= S_I \\ V_O(S_I) &= V_I(S_I). \end{aligned}$$

7. Set output binary matrix cells with objects-growing DM cells and assign the most informative velocity estimates (object growing ones; overwrite eventual existing velocities):
$$\begin{aligned} D_O^+ &= D_O \cup S_G \\ V_O(S_G) &= V_G(S_G). \end{aligned}$$

8. Clear cells with existing features:
$$D_O^+ = D_O \cap \overline{F_O}.$$

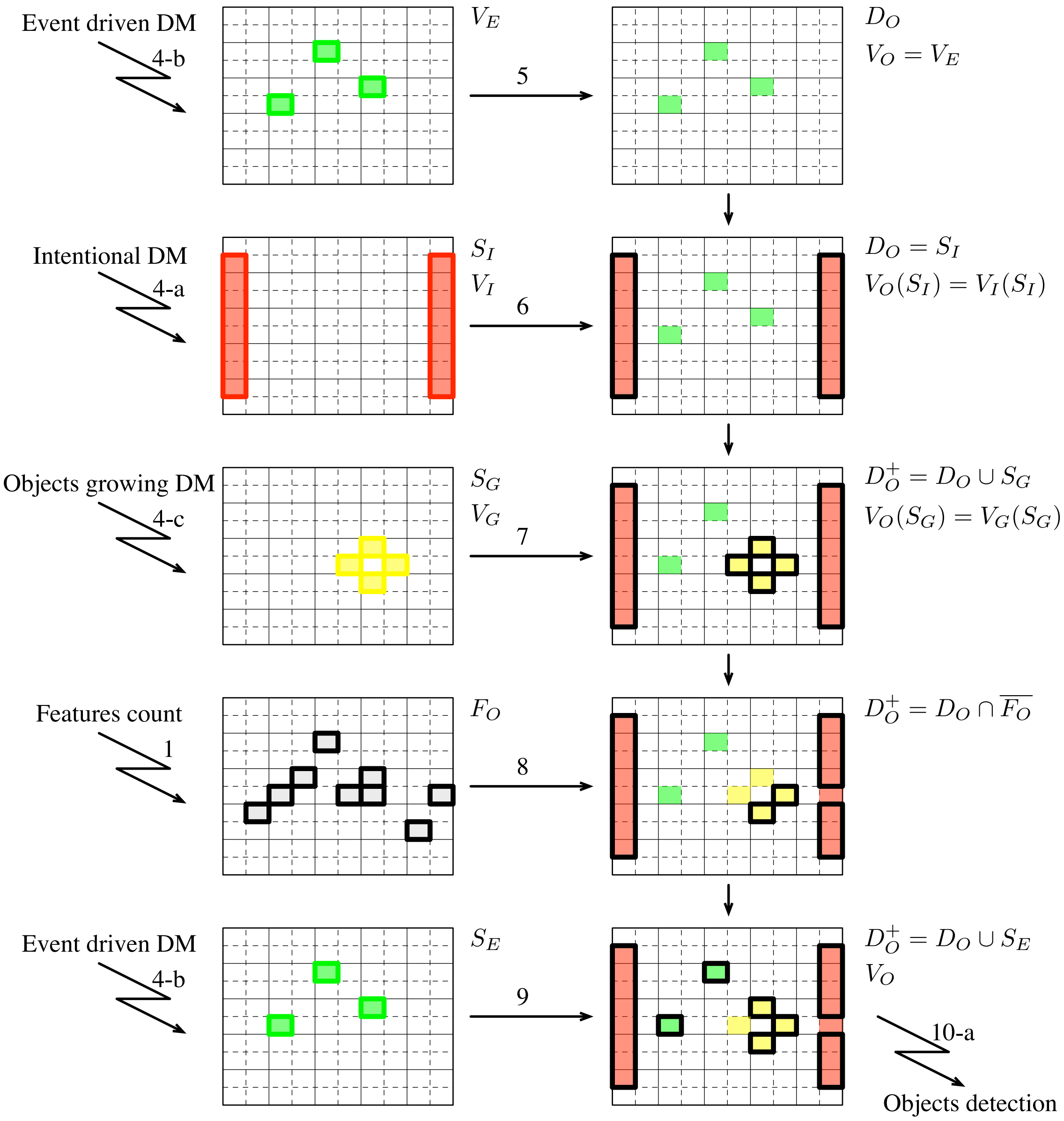

Figure 8.12: Priority-based Detection Selector algorithm -II-: Objects detection selector mechanism. The results of the different DM (*left*) sequentially update the output binary and velocity matrices D_O and V_O (*right*). A colored background indicates the nature of the velocity *a-priori* that has been set for that cell in V_O. Active cells in D_O are black-framed. Observe how the result (*bottom-right*) is a set of active cells with associated velocity *a-prioris* which respects the three imposed criterions.

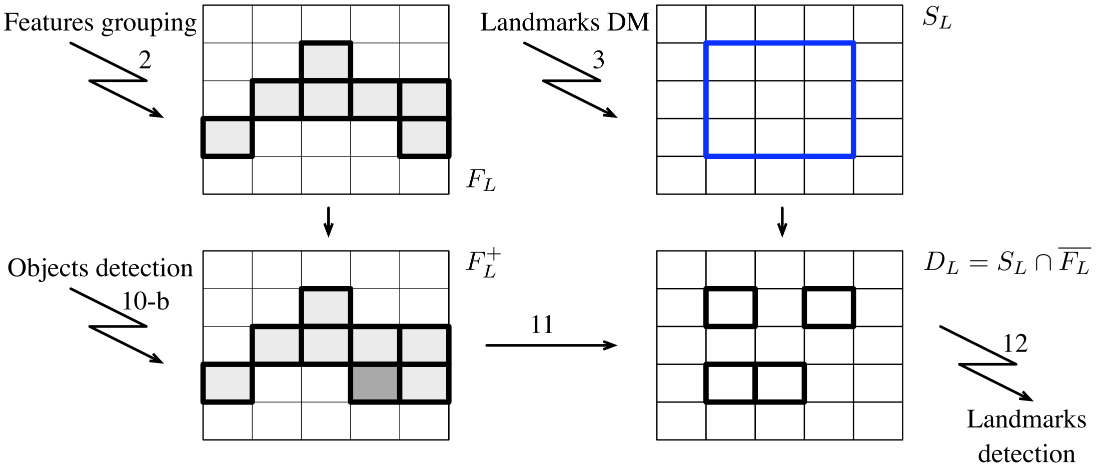

Figure 8.13: Priority-based Detection Selector algorithm -III-: Landmarks detection selector mechanism. A successful object creation (example) has set the *dark gray* cell in the features matrix F_L. On *bottom-right*, only cells marked with thick black frames will be evaluated for landmark detection.

9. Set event-driven cells (they must always be searched):

$$D_O^+ = D_O \cup S_E.$$

10. Perform feature detection in those image cells indicated by true entries in D_O. On successful detection:

    (a) create new moving objects and assign them the *a-priori* velocity means and covariances matrices contained in the corresponding entries of V_O;

    (b) set the appropriate cells in F_L to be non-empty.

11. Clear those landmarks detection cells which contain at least one feature. Assign to landmarks output binary matrix D_L:

$$D_L = S_L \cap \overline{F_L}.$$

12. Perform feature detection and landmark initializations in those cells indicated by true entries in D_L.

8.6 Some results

Results for this chapter are preliminary. The filtering side has been completely tested and tracking is successfully achieved. The perception side is still under construction: from the three proposed Detection Mechanisms, only Intentional DM has been coded and tested. Apart from these missing aspects, the system uses the full capabilities of the solutions developed so far: BiCam SLAM operation with extrinsic self-calibration and detection and tracking of moving objects.

We demonstrate BiCamSLAM-MOT with an indoor sequence taken by a forward-moving robot in the presence of a moving object traversing the scene from right to left. Two representative snapshots of this sequence are shown in Fig. 8.14.

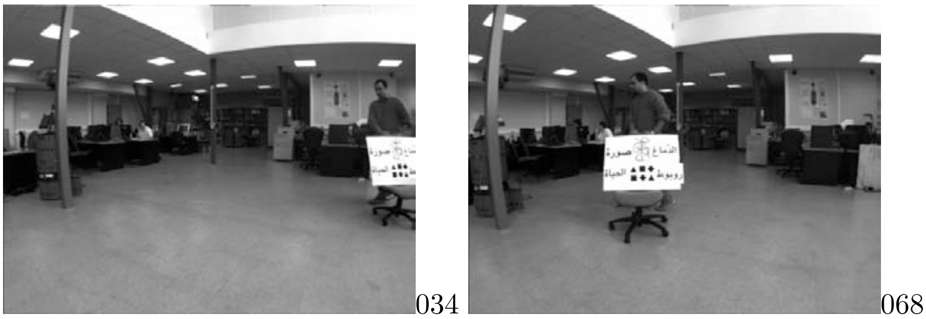

Figure 8.14: Two snapshots of the sequence.

We have used 6×6 and 12×12 cells grids for landmarks and objects detection respectively. The landmarks- and intentional- binary matrices are illustrated in Fig. 8.15. The estimates used for velocity priors and objects motion perturbations are summarized in Table 8.4. The rest of the parameters are exactly the same as those of the 'White board' experiments for the MonoSLAM and BiCamSLAM algorithms (Tables 6.3 and 7.1). Observe that velocity prior means and variances and velocity perturbations have very small vertical components (Z-axis). This reflects the knowledge we have about expected moving objects. The velocity priors, which can be resumed in the scalar case to $\{\bar{v}, \sigma_v\} = \{0.5, 0.5\}$m/s, are set to walking speeds, with uncertainties that cover from steady or backwards motion to forward motions slightly above 1m/s (recall we use 3σ ellipses for active search matching, *e.g.* we get $v_{bck} = \bar{v} - 3\,\sigma_V = -1$m/s and $v_{fwd} = \bar{v} + 3\,\sigma_v = 2$m/s).

We show further snapshots of the running algorithm in Fig. 8.16. Processing starts at frame 30, and one of every seven frames are shown. We highlight the following capabilities being successfully and simultaneously achieved:

- The system started with the stereo rig uncalibrated and the first landmarks are initialized in the form of rays; after just a few frames full 3D observability covers the whole experiment room.

- Observe how, inside the Intentional DM region, detected features are systematically assigned to moving objects. Some of these features correspond to static 3D points and are included into the SLAM map after velocity has converged to close to zero values.

- Effectively moving objects are detected and tracked until they abandon the image on the left-hand side. The addition of Objects-growing DM should make it possible to achieve a denser representation of the solid corresponding to the set of moving points. The

Table 8.4: Parameters for the SLAM-MOT experiment.

Parameter	Symbol		Value	Comment
Sampling time	T_s	=	200ms	
Inten. DM vel. mean	$\bar{\mathbf{v}}$	=	$[0.0,\ 0.5,\ 0.0]^\top m/s$	Robot frame
Inten. DM vel. variances	$\text{diag}(\mathbf{P_{vv}})$	=	$[0.5^2,\ 0.5^2,\ 0.05^2] m^2/s^2$	Robot frame
Perturbation noise	$\text{diag}(\mathbf{Q})$	=	$[0.2^2,\ 0.2^2,\ 0.05^2] m^2/s^2$	Robot frame

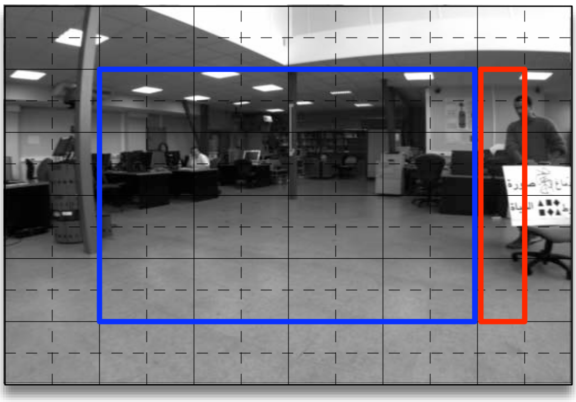

Figure 8.15: Detection Mechanisms: Landmarks (*blue*) and Intentional (*red*).

addition of the Event-driven DM should make it possible to detect the object occluding the landmarks at the end of the room (frames 58 to 65).

- Landmarks are initialized with lower density by using a thicker landmarks DM grid.

The full sequence can be better appreciated in movie format. For this, you can refer to the movies site `http://www.laas.fr/~jsola/objects/videos/PhD/video-8N.mov` , where `N` is a video number, or to the included CD.

8.7 Conclusions and further work

We have presented a real-time solution to the SLAM problem with Moving Objects Tracking that is exclusively based in visual exteroceptive perception. The tracking side of the problem has been shown to be not so fundamental as it could initially be thought because we could re-use the whole SLAM methods developed so far. On the contrary, the detection side of the problem is: the constraints imposed by limited computer resources required a clever selection of the interesting regions in the image in order to minimize the risks of missing newly appearing moving objects. We created the Detection Mechanisms to anticipate eventual object apparitions in the next few frames. These mechanisms had to be executed at disparate locations in the main SLAM filter loop. To correctly unify the action of all of them, some priority-based performance criterions have been announced and an algorithm to fulfill them has ben finally proposed. Preliminary results show that these decisions were on the right way.

Immediate further work is therefore to complete the experimentation in order to validate the proposed DM and/or imagine alternative ones. This work should be ready by the end of this thesis. Further reflections about this chapter should therefore wait for these works to be finished. For a more general discussion refer to the general conclusion that should come in a few pages.

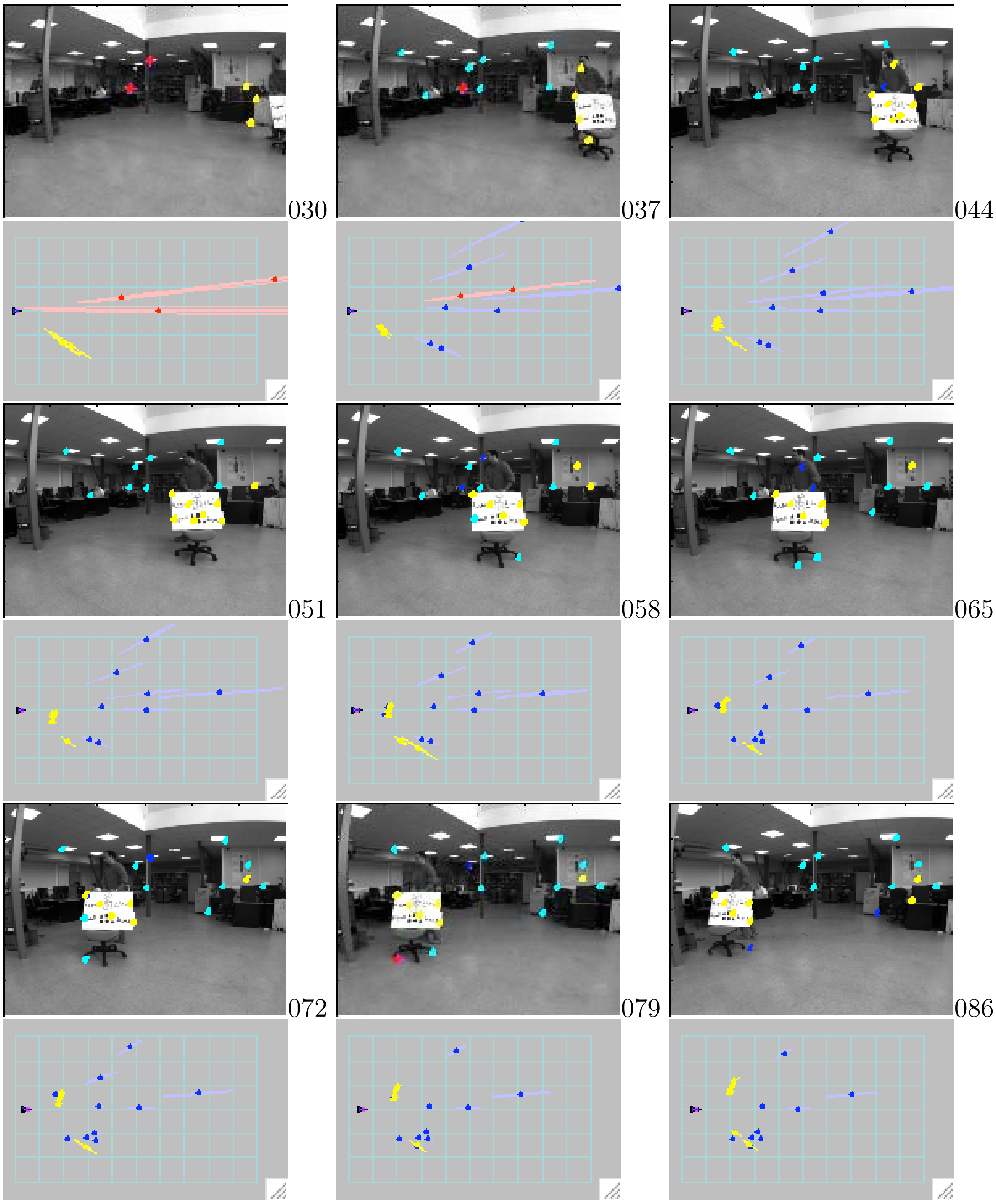

Figure 8.16: Nine snapshots of the image plane and top views of the produced 3D map. In the image view, *red* features are updated rays; *magenta* features are predicted rays (not updated); *cyan* features are updated landmarks; *blue* features are predicted landmarks; *yellow* features are updated moving objects; *orange* features are predicted objects. The mobile object in *yellow* has been successfully detected and tracked in 3D space as can be seen in the map views.

Conclusions

I have tried in this thesis to give new means for a machine to understand complex and dynamic visual scenes in real time. In particular, I solved the problem of simultaneously determining a certain world's geometry, the observer's trajectory, and the moving objects' structures and trajectories, with the aid of vision exteroceptive sensors. I proceeded by dividing the problem into three main steps: First, I gave a precursory solution to the SLAM problem for monocular vision that is able to adequately perform in the most ill-conditioned situations: those where the robot approaches the scene in straight line and where remote landmarks want to be taken into account. Second, I incorporated full 3D instantaneous observability by duplicating vision hardware with monocular algorithms. This permitted me to avoid some of the inherent drawbacks of classic stereo systems. Third, I added detection and tracking of moving objects by making use of this full 3D observability, whose necessity I judged almost inevitable. In order to alleviate the computational payload of the image processing algorithms that are required to extract the necessary geometrical information out of the images, I chose a sparse, punctual representation of both the world and the objects. This alleviation was additionally supported by active feature detection and search mechanisms which focus the attention to those image regions with the highest interest. This focusing was achieved by exploiting the current knowledge available on the system. The whole work was undertaken from a probabilistic viewpoint that fuses in real time information of exclusively geometric nature.

If I had to indicate the dark points of the chosen approaches, I would have to coincide with the recent arising voices claiming that punctual world representations cannot provide satisfactory mapping results. Indeed, the sparse set of estimated 3D points that constitutes the map is far from describing the complexity of the surrounding world. In particular, man-made indoor scenarios present relatively large non-textured surfaces that prevent any point-feature detected landmark to be associated. This means that a door, for instance, can be represented by its four corners, but we can never know whether it is open or closed.[12] Besides, outdoor scenarios do present variable textures but the perceived features are not very stable due to the fact that planar surfaces are rare.

Recent works on vision-based SLAM are already using segment-based landmarks which include a one-dimensional notion of continuity that provides the map with a much richer representativeness of reality. This is effectively an important step forward for structured scenarios with plenty of straight lines. I believe that the main concepts developed in this thesis (operation in ill-conditioned situations; multi-camera operation; priority-based detection mechanisms; active features search) can be directly applied to such kind of approaches.

[12] And the robot does not know either if those four points correspond to a door. Any concept like 'door', 'open' or 'closed' is very far from being deductible from purely geometrical information: some extra knowledge must be provided to the robot in advance.

Nevertheless, segments can not solve the question of the open or closed door either: a two-dimensional notion of continuity is necessary.

Texture- and color- based approaches provide this 2D continuity and would naturally make the concepts of 'surface' and 'solid' arise. Their difficulty of implementation in real-time systems is that they demand an exhaustive analysis of region-confined image properties.

With respect to the conic ray and its multi-Gaussian representation, I strongly believe that the technique can now be substituted by the inverse-depth parametrization recently appeared, which can be more theoretically defended. Besides, this parametrization is naturally adapted to landmarks depths reaching the infinity, thus truly incorporating stable angular references such as remote mountain peaks, stars, lighthouses or whatsoever. I also believe that this representation should eliminate the consistency problems that the self-calibration solution was showing.

A more difficult, challenging initiative would be to insist in obtaining a monocular solution to the Moving Objects Tracking problem. I already showed in the Introduction, with a simple video game example, that this should be possible, with the condition of incorporating other than geometrical considerations: besides algorithmic improvements, I honestly believe that the geometrical properties of the system cannot be pushed much further. Although it could be started simple, such a project should ultimately consider the following two issues: *a)* obtaining the most out of the observability properties of nearby structured moving objects, and *b)* dealing with remote moving objects. None of these issues seems easy to solve. Regarding observability of close objects, I indicated in the last chapter some of the techniques that could be used. However, I still think that this is not enough: those techniques are still based on a punctual representation of reality and thus suffer from important losses of information. With respect to remote objects, in my opinion the only way to geometrically solve the problem will be by exploiting the inverse relationship between the apparent object's size and its distance, something that requires the previous knowledge of the object size and thus reliable mechanisms for automatic recognition. More advanced techniques should try to recognize the landscape surfaces, with its intricate perspectives, to be able to locate distant objects that transit usually on ground. Once we start recognizing the vast set of techniques that we living beings use to understand a visual scene we easily block on panic: shadows, air transparency, previously known objects, previously known dynamics, unconsciously learned physics rules like the effect of gravity, friction or contact, reflections and refraction, etc, all play their roles in helping to compose a coherent representation of a dynamic, three-dimensional reality.

This brings the vision problem very close to a complete Artificial Intelligence (AI) problem, which in my opinion cannot be solved with one sole kind of approach. That is to say, there will not be *one* algorithm that performs this task, because the task is intricate with other parallel difficulties, and the required information and knowledge comes from a vast variety of sources. My general idea for conceiving algorithms for complex AI problems is sketched in Fig. 8.17: A parallel, independent set of processes is launched to analyze those aspects of the perceived reality which are considered relevant for their particular purposes. Each one of these processes delivers an output. A final fusion mechanism is the responsible of evaluating all outputs, resolving eventual conflicts and merging complementary parts to produce an adequate answer to the original stimuli. Such a scheme is used in voting-like algorithms, for instance, and is also the main concept I used for moving objects detection via independent Detection Mechanisms and the Priority-Based Detection Selector. All these procedures must

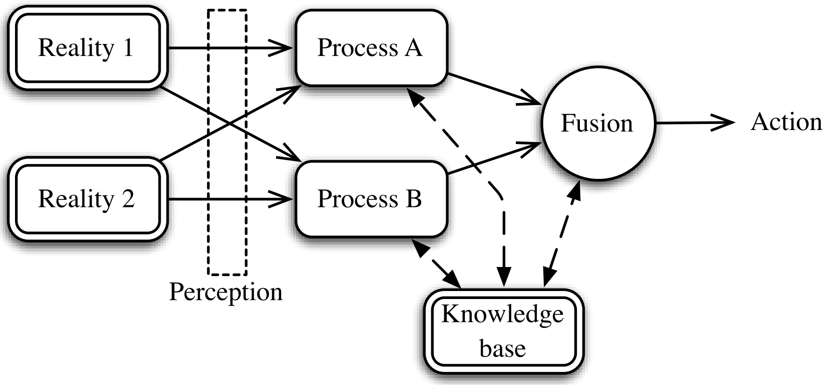

Figure 8.17: Parallel processing and late fusion.

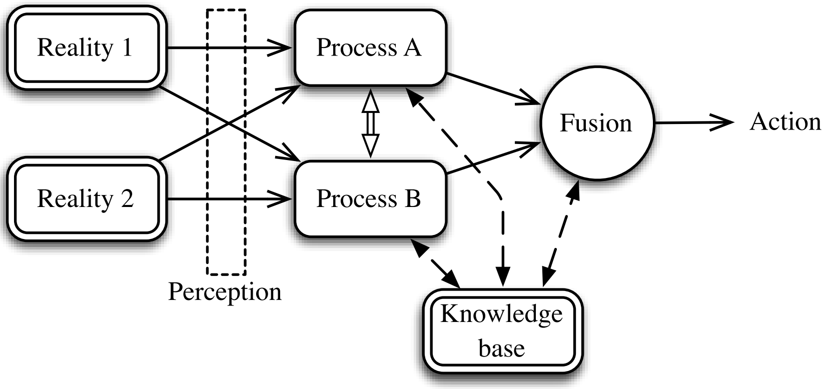

Figure 8.18: Collaborative processing and late fusion.

be effectively driven by a Knowledge Base, which may be able to focus the operation to the most interesting data, and which is continuously augmented and refined by the system (this would be the map in the SLAM case).

The performances of such a scheme could be significantly improved by adding algorithm collaboration as in Fig. 8.18. That is, before each independent procedure delivers its output, it can already give (and receive) important clues to (and from) its colleagues. This requires establishing a common intermediary language between processes so that these information can be correctly interpreted and successfully incorporated as valuable new inputs.

By considering exclusively a forward propagation of the information within the processing algorithms, the above collaboration can be implemented by a neural-network-like structure (Fig. 8.19). The fundamental difference with neural networks is the fact that each node is not a simple 'rather silly' operator such as a neuron but a complete and complex algorithm.

Finally and going back to our problem, I beg for your consideration to let me adventure the following proposition. A collaborative-like approach could be imagined for a monocular solution to the MOT problem as follows (Fig. 8.20): Take visual reality and divide it into color and luminance. Take a point feature detector and a color segmentation algorithms. Group the outputs of the feature detector as sets of points corresponding to the same color region. Assign this way *a-priori* feature-object associations. Run a RANSAC-based solution to solve each

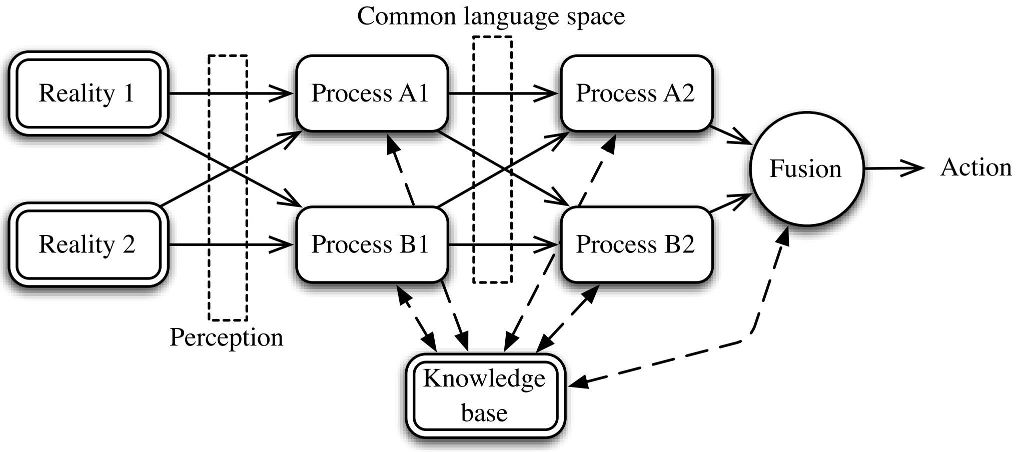

Figure 8.19: Neural-like processing and late fusion.

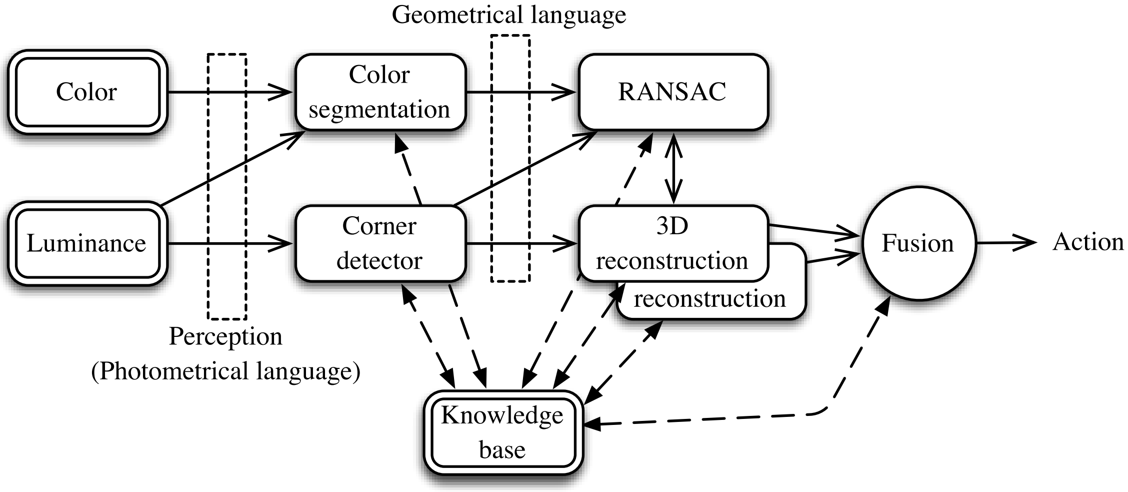

Figure 8.20: A simplified AI knowledge flow graph for MOT using monocular vision.

camera-object pair as in Section 8.3.1 with reduced dimensionality of the search space. Solve each 3D reconstruction problem and unify camera motion to get a coherent, single referenced world, camera and moving objects representation.

I am not saying that the problem is solvable this way, but the concept of parallel, collaborative processes seems to me an appealing starting point to imagine really intelligent artificial processing: they provide decentralized operation which can be easily paralleled; they are knowledge based which eases-up fulfilling real-time constraints; and they share preliminary information so that all the paralleled agents benefit each other.

As a concluding remark, I would like to mention that in the course of this work I have realized that… I am more interested in intelligence than vision. Indeed, I took the whole vision problematic as an exercise to '*reason about reasoning*'. By now, I am not sure where this deep trend will lead me. It is almost sure that I do not pursuit my career in the vision community, at least if this means continuing to code single-task algorithms. I am not a good code-writer anyway. However, my ‘engineering’ background makes it not so easy to switch into more fundamental disciplines such as Mathematics or Physics, where my preferences for deep philosophical insights would find perhaps more fertilized grounds.

I’m still searching. . .

Part III

Appendices

Appendix A

Differentiation of matrices and vectors

A.1 Differentiation with respect to a scalar

We define differentiation of a vector or matrix with respect to a scalar in the following way:

$$\mathbf{z} = \mathbf{z}(t), \qquad \frac{d\mathbf{z}}{dt} \triangleq \begin{bmatrix} \frac{dz_1}{dt} & \frac{dz_2}{dt} & \cdots & \frac{dz_n}{dt} \end{bmatrix}^\top \tag{A.1}$$

$$\mathbf{F} = \mathbf{F}(t), \qquad \frac{d\mathbf{F}}{dt} \triangleq \begin{bmatrix} \frac{df_{11}}{dt} & \frac{df_{12}}{dt} & \cdots & \frac{df_{1n}}{dt} \\ \frac{df_{21}}{dt} & \frac{df_{22}}{dt} & \cdots & \frac{df_{2n}}{dt} \\ \vdots & \vdots & & \vdots \\ \frac{df_{m1}}{dt} & \frac{df_{m2}}{dt} & \cdots & \frac{df_{mn}}{dt} \end{bmatrix} \tag{A.2}$$

A.2 Differentiation with respect to a vector

Differentiation of a scalar f with respect to a vector $\mathbf{x}$ is often called the gradient and written $\nabla_{\mathbf{x}} f = \frac{\partial f}{\partial \mathbf{x}}$. It is defined in the literature in two different ways[1], depending on the orientation of the obtained vector, as follows:

$$\begin{aligned} \nabla_{\mathbf{x}} f\big|_1 &\doteq \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix} \\ \nabla_{\mathbf{x}} f\big|_2 &\doteq \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^\top \end{aligned}$$

To clarify things up, we will explicitly denote the orientation of the vector with respect to which we differentiate. Consider then the following definitions:

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{x}} &\triangleq \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}^\top \\ \frac{\partial f}{\partial \mathbf{x}^\top} &\triangleq \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}. \end{aligned} \tag{A.3}$$

Notice that the differentiator's orientation drives that of the output vector.

[1] Indeed without any kind of consensus. In `www.wikipedia.org`, for example, the gradient of a scalar $f(\mathbf{x})$ with respect to vector $\mathbf{x}$ is defined as a *column* vector; later, they define the Jacobian matrix of the vector function $\mathbf{f}(\mathbf{x})$ and they say its *rows* are the gradients of each scalar component of $\mathbf{f}$!

Similarly for vector functions $\mathbf{z} = \mathbf{z}(\mathbf{x})$, we have two alternatives[2]. Observe how both the differentiated- and the differentiator-vector orientations drive the distribution of the entries in the output matrices:

$$\frac{\partial \mathbf{z}^\top}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_1} \\ \frac{\partial z_1}{\partial x_2} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_m}{\partial x_2} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_1}{\partial x_n} & \frac{\partial z_2}{\partial x_n} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix}$$

and

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top} \triangleq \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \cdots & \frac{\partial z_1}{\partial x_n} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_m}{\partial x_1} & \frac{\partial z_m}{\partial x_2} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix} \tag{A.4}$$

These forms are called the Jacobian matrices (usually written $J_\mathbf{x}\mathbf{z}(\mathbf{x})$ or $\nabla_\mathbf{x}\mathbf{z}$). The choice of the forms (A.3) and (A.4), with the transposed differentiator, will become handy when we introduce the partial vector and matrix derivatives in A.3: the rules will be almost equivalent to those known for the scalar case, thus much more intuitive and easy to remember. We will systematically use the second forms. Further, for the Jacobian matrices evaluated at a certain point $\mathbf{x}_0$ we will adopt the notation

$$\mathbf{z} = \mathbf{z}(\mathbf{x}); \qquad \mathbf{Z}_\mathbf{x} \triangleq \left.\frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top}\right|_{\mathbf{x}_0} . \tag{A.5}$$

The derivative of a matrix with respect to a vector is here defined as a special partitionned matrix

$$\mathbf{F} = \mathbf{F}(\mathbf{x})$$

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}^\top} \triangleq \left[\frac{\partial \mathbf{F}}{\partial x_1} \vdots \frac{\partial \mathbf{F}}{\partial x_2} \vdots \ldots \vdots \frac{\partial \mathbf{F}}{\partial x_n} \right] \tag{A.6}$$

A.3 Operations involving partial derivatives

Working with the forms with transposed differentiators leads to a natural extension of the derivation and partial-derivation rules for composed functions. Here only this form is shown.

Consider the functions

$$f = f(\mathbf{y}, \mathbf{x}, t) \qquad \mathbf{y} = \mathbf{y}(\mathbf{x}, t) \qquad \mathbf{x} = \mathbf{x}(t)$$

[2] We have actually four, but $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}^\top}{\partial \mathbf{x}^\top}$ are not Jacobian matrices but vectors. They are not really useful after all.

The following properties hold:

$$\frac{\partial f}{\partial \mathbf{x}^\top} = \frac{\partial f}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial f}{\partial \mathbf{x}^\top} \tag{A.7}$$

$$\frac{\partial f}{\partial t} = \left(\frac{\partial f}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial f}{\partial \mathbf{x}^\top}\right)\frac{\partial \mathbf{x}}{\partial t} + \frac{\partial f}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial t} + \frac{\partial f}{\partial t} \tag{A.8}$$

Similar operations on the vector functions

$$\mathbf{z} = \mathbf{z}(\mathbf{y}, \mathbf{x}, t) \qquad \mathbf{y} = \mathbf{y}(\mathbf{x}, t) \qquad \mathbf{x} = \mathbf{x}(t)$$

are defined by

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top} = \frac{\partial \mathbf{z}}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top} \tag{A.9}$$

$$\frac{\partial \mathbf{z}}{\partial t} = \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top}\right)\frac{\partial \mathbf{x}}{\partial t} + \frac{\partial \mathbf{z}}{\partial \mathbf{y}^\top}\frac{\partial \mathbf{y}}{\partial t} + \frac{\partial \mathbf{z}}{\partial t} \tag{A.10}$$

A.4 Linear forms

It will be convenient to use the concept of expanded matrices in what is to follow. We define

$$[\mathbf{I}]_{\mathbf{A}} \triangleq \begin{bmatrix} \mathbf{A} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \cdots & \mathbf{0} \\ \vdots & & & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{A} \end{bmatrix} \qquad [\mathbf{A}]_{\mathbf{I}} \triangleq \begin{bmatrix} a_{11}\mathbf{I} & a_{12}\mathbf{I} & \cdots & a_{1n}\mathbf{I} \\ a_{21}\mathbf{I} & a_{22}\mathbf{I} & \cdots & a_{2n}\mathbf{I} \\ \vdots & & & \vdots \\ a_{m1}\mathbf{I} & a_{m2}\mathbf{I} & \cdots & a_{mn}\mathbf{I} \end{bmatrix} \tag{A.11}$$

which is a Kronecker product type construction valid regardless of the dimension of $\mathbf{A}$.

If, for instance,

$$\mathbf{z} = \mathbf{A}\mathbf{y}, \qquad \mathbf{A} = \mathbf{A}(\mathbf{x}, t), \qquad \mathbf{y} = \mathbf{y}(\mathbf{x}, t), \qquad \mathbf{x} = \mathbf{x}(t)$$

then

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}^\top} = \mathbf{A}\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top} + \frac{\partial \mathbf{A}}{\partial \mathbf{x}^\top}[\mathbf{I}]_{\mathbf{y}} \tag{A.12}$$

and

$$\frac{\partial \mathbf{z}}{\partial t} = \mathbf{A}\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}^\top}\frac{\partial \mathbf{x}}{\partial t} + \frac{\partial \mathbf{y}}{\partial t}\right) + \left(\frac{\partial \mathbf{A}}{\partial \mathbf{x}^\top}\left[\frac{\partial \mathbf{x}}{\partial t}\right]_{\mathbf{I}} + \frac{\partial \mathbf{A}}{\partial t}\right)\mathbf{y} \tag{A.13}$$

A.5 Local function linearization

The generalized Taylor expansion for non-linear, m-dimensional functions of n-dimensional vectors $\mathbf{f}(\mathbf{x})$ around the support point $\mathbf{x}_0$ is written as:

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top}\right|_{\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + \text{higher-order terms.} \tag{A.14}$$

When truncated just after the linear term, the Taylor series is the best linear approximation of $\mathbf{f}(\mathbf{x})$ around $\mathbf{x}_0$ in the least squares sense:

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_0) + \mathbf{F}_\mathbf{x}(\mathbf{x} - \mathbf{x}_0) \tag{A.15}$$

where $\mathbf{F}_\mathbf{x}$ is the Jacobian matrix

$$\mathbf{F}_\mathbf{x} \triangleq \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top} \right|_{\mathbf{x}_0} .$$

A.6 Modular computation of Jacobian matrices.

In this thesis we deal with complex non-linear functions that require numerous successive linearizations at each estimate update. The computation of these Jacobian matrices is performed by functions that contain in its code their analytic expressions and that take the evaluation point as the input parameter. The obtaining of the analytic expressions is normally commissioned to automatic symbolic calculators such as Matlab or MAPPLE (the former is actually utilizing MAPPLE libraries) which often return huge and cryptic results. The final user rarely attempts to understand them.

We show that Jacobian expressions can be elucidated by considering the elementary functions: the Jacobian matrices of the composed functions are just proper combinations of those of the elementary functions.

A.6.1 Rules of composition

Jacobian composition just mimics the rules of the partial derivatives in the scalar case. We see this with an example. Consider the non-linear functions

$$\mathbf{u} = \mathbf{f}(\mathbf{x}, \mathbf{y}) \qquad \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{z})$$

and their elementary Jacobian matrices defined as functions of their linearization points as

$$\begin{aligned}
\mathbf{F}_\mathbf{x}(\mathbf{x}_0, \mathbf{y}_0) &= \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top} \right|_{\mathbf{x}_0, \mathbf{y}_0} \\
\mathbf{F}_\mathbf{y}(\mathbf{x}_0, \mathbf{y}_0) &= \left. \frac{\partial \mathbf{f}}{\partial \mathbf{y}^\top} \right|_{\mathbf{x}_0, \mathbf{y}_0} \\
\mathbf{G}_\mathbf{x}(\mathbf{x}_0, \mathbf{z}_0) &= \left. \frac{\partial \mathbf{g}}{\partial \mathbf{x}^\top} \right|_{\mathbf{x}_0, \mathbf{z}_0} \\
\mathbf{G}_\mathbf{z}(\mathbf{x}_0, \mathbf{z}_0) &= \left. \frac{\partial \mathbf{g}}{\partial \mathbf{z}^\top} \right|_{\mathbf{x}_0, \mathbf{z}_0}
\end{aligned}$$

From the generalization of A.9, the Jacobian matrices of the composed function

$$\mathbf{u} = \mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{f}\big[\mathbf{x}\, ,\, \mathbf{g}(\mathbf{x}, \mathbf{z})\big]$$

with respect to $\mathbf{x}$ and $\mathbf{z}$ evaluated around the linearization points $\mathbf{x}_0$ and $\mathbf{z}_0$ are obtained with the products of the elementary Jacobian matrices as follows: first compute the intermediate linearization point

$$\mathbf{y}_0 = \mathbf{g}(\mathbf{x}_0, \mathbf{z}_0) \tag{A.16}$$

and then

$$\mathbf{H_x}(\mathbf{x}_0,\mathbf{z}_0) \triangleq \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}^\top}\right|_{\mathbf{x}_0,\mathbf{z}_0} = \mathbf{F_y}(\mathbf{x}_0,\mathbf{y}_0)\cdot\mathbf{G_x}(\mathbf{x}_0,\mathbf{z}_0) + \mathbf{F_x}(\mathbf{x}_0,\mathbf{y}_0) \tag{A.17}$$

$$\mathbf{H_z}(\mathbf{x}_0,\mathbf{z}_0) \triangleq \left.\frac{\partial \mathbf{h}}{\partial \mathbf{z}^\top}\right|_{\mathbf{x}_0,\mathbf{z}_0} = \mathbf{F_y}(\mathbf{x}_0,\mathbf{y}_0)\cdot\mathbf{G_z}(\mathbf{x}_0,\mathbf{z}_0). \tag{A.18}$$

A.6.2 Jacobian matrices of some elementary functions

From the whole set of elementary functions we are dealing with, there are two which merit a special attention: $fromFrame()$ and $toFrame()$. In these functions the derivation rules and definitions given so far are not sufficient to obtain a satisfactory closed, compact formulation for their Jacobian matrices. The missing steps are thus given here.

We give the Jacobians of functions $toFrame(\cdot)$ and $fromFrame(\cdot)$ when orientations are specified in quaternions. The specification of a generic frame $\mathcal{F}$ with respect to another one $\mathcal{W}$ is expressed as a 7-dimension vector:

$$\mathcal{F} = \begin{bmatrix}\mathbf{t}\\ \mathbf{q}\end{bmatrix} = \begin{bmatrix}t_x & t_y & t_z & a & b & c & d\end{bmatrix}^\top. \tag{A.19}$$

The functions $fromFrame()$ and $toFrame()$ become:

$$\begin{aligned} fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}}) &= \mathbf{R}(\mathbf{q})\,\mathbf{p}^{\mathcal{F}} + \mathbf{t} \\ toFrame(\mathcal{F},\mathbf{p}^{\mathcal{W}}) &= \mathbf{R}^\top(\mathbf{q})\,\mathbf{p}^{\mathcal{W}} - \mathbf{R}(\mathbf{q})^\top\,\mathbf{t} \end{aligned}$$

where we recall that $\mathbf{R}^\top(\mathbf{q}) = \mathbf{R}(\mathbf{q}^*)$. Their Jacobians with respect to frame $\mathcal{F}$ are then block-defined from the Jacobians with respect to $\mathbf{t}$ and $\mathbf{q}$:

$$\begin{aligned} \mathbf{FF}_{\mathcal{F}} &= \begin{bmatrix}\mathbf{FF_t} \; \mathbf{FF_q}\end{bmatrix} \\ \mathbf{TF}_{\mathcal{F}} &= \begin{bmatrix}\mathbf{TF_t} \; \mathbf{TF_q}\end{bmatrix} \end{aligned}$$

fromFrame()

The Jacobians of $fromFrame(\cdot)$ are defined by:

$$\begin{aligned} \mathbf{FF_t}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\, fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{t}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} = \mathbf{I}_{3\times 3} \\ \mathbf{FF_q}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\, fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{q}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} = \left.\frac{\partial(\mathbf{R}(\mathbf{q})\cdot\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{q}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} \\ \mathbf{FF_p}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) &\triangleq \left.\frac{\partial\, fromFrame(\mathcal{F},\mathbf{p}^{\mathcal{F}})}{\partial \mathbf{p}^{\mathcal{F}\top}}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} = \mathbf{R}(\hat{\mathbf{q}}) \end{aligned}$$

where, using (A.12), $\mathbf{FF_q}$ can be written as

$$\mathbf{FF_q}(\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}) = \begin{bmatrix}\frac{\partial \mathbf{R}}{\partial a}\mathbf{p}^{\mathcal{F}} & \frac{\partial \mathbf{R}}{\partial b}\mathbf{p}^{\mathcal{F}} & \frac{\partial \mathbf{R}}{\partial c}\mathbf{p}^{\mathcal{F}} & \frac{\partial \mathbf{R}}{\partial d}\mathbf{p}^{\mathcal{F}}\end{bmatrix}_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{F}}} \tag{A.20}$$

which is difficult to further develop in the general case. The quaternion representation allows for more interesting, compact expressions, which are given now. A specific development of (A.20) using the quaternion-based definition of the rotation matrix (Appendix C) leads to the following procedure: build the matrix

$$\mathbf{\Pi}(\hat{\mathbf{q}}) = \begin{bmatrix} -b & -c & -d \\ a & -d & c \\ d & a & -b \\ -c & b & a \end{bmatrix} \tag{A.21}$$

then the vector

$$\mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = 2\mathbf{\Pi}\,\hat{\mathbf{p}}^{\mathcal{F}} \tag{A.22}$$

and the Jacobian is finally

$$\mathbf{FF_q}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{F}}) = \begin{bmatrix} s_2 & -s_1 & s_4 & -s_3 \\ s_3 & -s_4 & -s_1 & s_2 \\ s_4 & s_3 & -s_2 & -s_1 \end{bmatrix}. \tag{A.23}$$

toFrame()

The Jacobians of $toFrame(\cdot)$ are defined by:

$$\begin{aligned}
\mathbf{TF_t}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{W}}) &\triangleq \left.\frac{\partial\; toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}})}{\partial \mathbf{t}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{W}}} = -\mathbf{R}^\top(\hat{\mathbf{q}}) \\
\mathbf{TF_q}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{W}}) &\triangleq \left.\frac{\partial\; toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}})}{\partial \mathbf{q}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{W}}} = \left.\frac{\partial(\mathbf{R}^\top(\mathbf{q}) \cdot (\mathbf{p}^{\mathcal{W}} - \mathbf{t}))}{\partial \mathbf{q}^\top}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{W}}} \\
\mathbf{TF_p}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{W}}) &\triangleq \left.\frac{\partial\; toFrame(\mathcal{F}, \mathbf{p}^{\mathcal{W}})}{\partial \mathbf{p}^{\mathcal{W}\top}}\right|_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{W}}} = \mathbf{R}^\top(\hat{\mathbf{q}})
\end{aligned}$$

where $\mathbf{TF_q}$ can be further developed as

$$\mathbf{TF_q}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{W}}) = \begin{bmatrix} \frac{\partial \mathbf{R}^\top}{\partial a}(\mathbf{p}^{\mathcal{W}} - \mathbf{t}) & \frac{\partial \mathbf{R}^\top}{\partial b}(\mathbf{p}^{\mathcal{W}} - \mathbf{t}) & \frac{\partial \mathbf{R}^\top}{\partial c}(\mathbf{p}^{\mathcal{W}} - \mathbf{t}) & \frac{\partial \mathbf{R}^\top}{\partial d}(\mathbf{p}^{\mathcal{W}} - \mathbf{t}) \end{bmatrix}_{\hat{\mathbf{t}},\hat{\mathbf{q}},\hat{\mathbf{p}}^{\mathcal{W}}}$$

To obtain the closed forms build the matrix

$$\mathbf{\Pi}^*(\hat{\mathbf{q}}) = \mathbf{\Pi}(\hat{\mathbf{q}}^*) = \begin{bmatrix} b & c & d \\ a & d & -c \\ -d & a & b \\ c & -b & a \end{bmatrix} \tag{A.24}$$

then the vector

$$\mathbf{s}^* = \begin{bmatrix} s_1^* \\ s_2^* \\ s_3^* \\ s_4^* \end{bmatrix} = 2\mathbf{\Pi}^*(\hat{\mathbf{p}}^{\mathcal{W}} - \hat{\mathbf{t}}) \tag{A.25}$$

and the Jacobian is finally

$$\mathbf{TF_q}(\hat{\mathbf{t}}, \hat{\mathbf{q}}, \hat{\mathbf{p}}^{\mathcal{W}}) = \begin{bmatrix} s_2^* & s_1^* & -s_4^* & s_3^* \\ s_3^* & s_4^* & s_1^* & -s_2^* \\ s_4^* & -s_3^* & s_2^* & s_1^* \end{bmatrix} \tag{A.26}$$

Appendix B

Facts on Gaussian variables

B.1 N-dimensional Gaussian random variables

The probability distribution function (*pdf*) of a n-dimensional, Gaussian-distributed random variable $\mathbf{x}$ is defined by

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}-\bar{\mathbf{x}};\mathbf{X}) \triangleq \frac{1}{\sqrt{(2\pi)^n|\mathbf{X}|}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\bar{\mathbf{x}})^\top\mathbf{X}^{-1}(\mathbf{x}-\bar{\mathbf{x}})\Big). \tag{B.1}$$

This *pdf* is fully specified by providing the vector $\bar{\mathbf{x}}$ and the matrix $\mathbf{X}$, its first- and second-order expectations respectively. In effect, applying the expectation operator we have

$$\begin{aligned} \bar{\mathbf{x}} &= E\left[\mathbf{x}\right] \\ \mathbf{X} &= E\left[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^\top\right] \end{aligned} \tag{B.2}$$

and we call $\bar{\mathbf{x}}$ the mean of $\mathbf{x}$ and $\mathbf{X}$ its covariances matrix. Notice that the covariances matrix $\mathbf{X}$ is by definition a square, symmetric and positive-defined matrix.

It is common to specify the *pdf* of a Gaussian variable with the notations

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}-\bar{\mathbf{x}};\mathbf{X}), \qquad p(\mathbf{x}) = \Gamma(\mathbf{x}-\bar{\mathbf{x}};\mathbf{X}),$$

where $\mathcal{N}$ states for 'normal distribution', an alternative denomination of the Gaussian distribution. In this document, when we just want to indicate the Gaussian character of a variable we simply use

$$\mathbf{x} \sim \mathcal{N}\{\bar{\mathbf{x}};\mathbf{X}\}$$

that clearly reads: *"$\mathbf{x}$ is Gaussian with mean $\bar{\mathbf{x}}$ and covariances matrix $\mathbf{X}$"*.

B.2 Ellipsoidal representation of Gaussians

In figures and movies, we usually represent Gaussian variables by ellipses and ellipsoids. This is because these geometric shapes are the boundaries of constant probability density that enclose a predefined region of confidence. We aim to obtain the geometric properties of these ellipsoids as a function of the Gaussian parameters $\bar{\mathbf{x}}$ (mean) and $\mathbf{X}$ (covariances matrix).

Consider the Gaussian variable $\mathbf{x} \sim \mathcal{N}\{\bar{\mathbf{x}};\mathbf{X}\}$. In its *pdf* expression (B.1) the term multiplying the exponential is just a *normalization factor*. Thus, to find the shape of constant

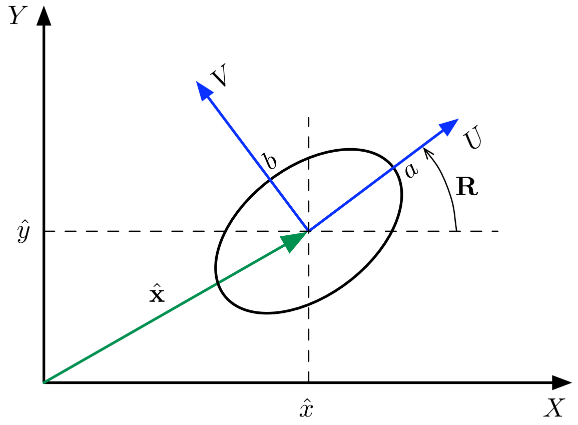

Figure B.1: Ellipse representing the 1σ bound region of confidence of a 2-dimensional Gaussian.

probability boundaries, consider only the exponent, and look for the geometric place of the points satisfying

$$(\mathbf{x}-\bar{\mathbf{x}})^\top \mathbf{X}^{-1}(\mathbf{x}-\bar{\mathbf{x}}) = 1. \tag{B.3}$$

Because $\mathbf{X}$ is symmetric and positive-defined, this is the equation of an n-dimensional ellipsoid (Fig. B.1 illustrates this fact in the 2D case). The ellipsoid's parameters are easily obtained from the singular value decomposition of $\mathbf{X}$ given by

$$\mathbf{X} = \mathbf{R}\mathbf{D}\mathbf{R}^\top \tag{B.4}$$

where $\mathbf{D}$ is a diagonal matrix containing the singular values of $\mathbf{X}$, and $\mathbf{R}$ is an orthonormal or rotation matrix. We remark that if we impose a particular ordering on the diagonal entries of $\mathbf{D}$ (for example a non-increasing order), this decomposition is unique. We give the following propositions:

Proposition B.1 (Ellipsoid center). *The ellipsoid's center is at the Gaussian's mean.* □

Proposition B.2 (Ellipsoid semi-axes). *The ellipsoid's semi-axes are the square roots of the singular values of the Gaussian's covariances matrix.* □

Proposition B.3 (Ellipsoid orientation). *The ellipsoid orientation is represented by the rotation matrix of the singular value decomposition of the Gaussian's covariances matrix.* □

Proof All propositions are proved by construction for the 2D case, the general n-D case being just an obvious extension. Consider an ellipse centered at the origin and aligned with the U and V coordinate axes (Fig. B.1), with semi-axes a and b satisfying $a \geq b$, and with equation $u^2/a^2 + v^2/b^2 = 1$. Write this equation in matrix form as

$$\begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 1 \tag{B.5}$$

Now define $\mathbf{D} = \operatorname{diag}(a^2, b^2)$ and $\mathbf{u} = [u, v]^\top$. Perform the Euclidean rigid frame transformation $\mathbf{x} = \mathbf{R}\mathbf{u} + \bar{\mathbf{x}}$ (Chapter 1) from axes UV to XY, which is driven by a translation $\bar{\mathbf{x}}$ and a

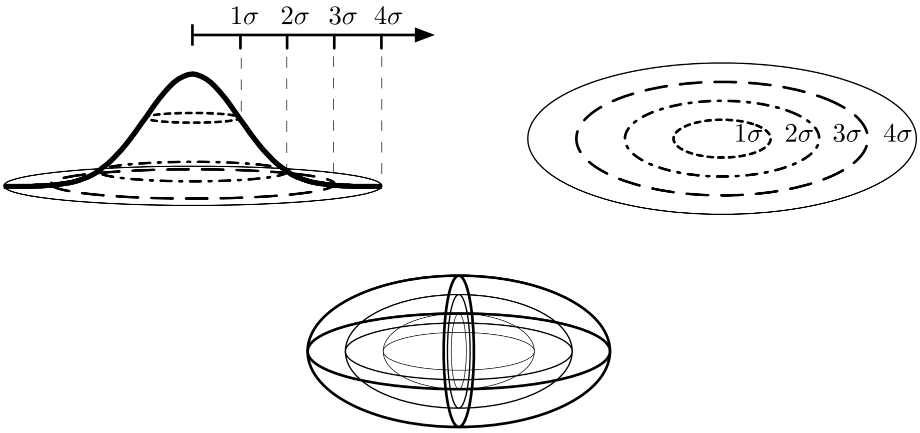

Figure B.2: 2D Gaussian distribution (*top*). Constant probability density planes cut the *pdf* in concentric ellipses (*left*), that can be classified with the number of standard deviations σ (*right*) or integer Mahalanobis distances. Three concentric ellipsoids (*bottom*) representing the 2σ, 3σ and 4σ equiprobable surfaces of a 3D Gaussian *pdf*.

rotation $\mathbf{R}$. Develop (B.5) to obtain $(\mathbf{x}-\bar{\mathbf{x}})^\top \mathbf{R}\mathbf{D}^{-1}\mathbf{R}^\top(\mathbf{x}-\bar{\mathbf{x}}) = 1$. This translates to

$$(\mathbf{x}-\bar{\mathbf{x}})^\top (\mathbf{R}\mathbf{D}\mathbf{R}^\top)^{-1}(\mathbf{x}-\bar{\mathbf{x}}) = 1$$

Finally consider $\mathbf{R}\mathbf{D}\mathbf{R}^\top$ as being the singular value decomposition of $\mathbf{X}$. The previous expression becomes (B.3) and thus the ellipse is conveniently represented by the Gaussian parameters. ■

Consider now the definition:

Definition B.1 (Mahalanobis distance). Given a multivariate Gaussian random vector $\mathbf{x} \sim \{\bar{\mathbf{x}}; \mathbf{X}\}$, the Mahalanobis distance from a point $\mathbf{x}$ to the mean $\bar{\mathbf{x}}$ is defined by

$$D_M = \sqrt{(\mathbf{x}-\bar{\mathbf{x}})^\top \mathbf{X}^{-1}(\mathbf{x}-\bar{\mathbf{x}})}. \qquad \square$$

It is clear that the ellipsoid (B.3) is the geometric place of the points at a unit Mahalanobis distance from the Gaussian mean. We call this ellipsoid the 1σ (one sigma) bound ellipsoid[1]. We can also construct bigger interesting ellipsoids, in particular the 2σ and the 3σ ones. In general, the $n\sigma$ ellipsoid is defined as the geometric place of the points at a Mahalanobis distance n from the Gaussian mean (Fig. B.2).

Integrating the Gaussian *pdf* inside these ellipsoids allows us to talk about regions of confidence. Table B.1 shows the probability of $\mathbf{x}$ being inside the $n\sigma$ ellipsoid, for $1 \leq n \leq 4$, in the 2D and 3D cases. Wee see that, in order to graphically represent Gaussians, it will be convenient to draw the 3σ bound ellipses and ellipsoids, as they enclose regions of confidence of about 98%.

[1] By analogy with scalar Gaussians where (B.3) becomes $x^2/\sigma^2 = 1$ and hence $x = 1\sigma$.

Table B.1: Percent probabilities of a random variable being inside its $n\sigma$ ellipsoid.

	1σ	2σ	3σ	4σ
2D	$39,4\%$	$86,5\%$	$98,9\%$	$99,97\%$
3D	$19,9\%$	$73,9\%$	$97,1\%$	$99,89\%$

B.3 Measures of uncertainty

Sometimes it may be useful to compare the uncertainty of two different Gaussian variables, in an easy and somewhat naive form, so as to be able to say "*A is more uncertain than B*". In the scalar case, one can just compare the variables standard deviations σ_A and σ_B and conclude that the above affirmation is true if and only if $\sigma_A > \sigma_B$.

For higher dimensional Gaussian variables $\mathbf{x} \sim \mathcal{N}\{\bar{\mathbf{x}}; \mathbf{X}\}$ a useful uncertainty measure is the volume of the corresponding ellipsoid: the bigger the volume, the more uncertain the variable is. Consider the following proposition:

Proposition B.4 (Ellipsoid volume). *The volume inside the 1σ bound n-ellipsoid corresponding to the n-dimensional Gaussian variable $\mathbf{x} \sim \mathcal{N}\{\bar{\mathbf{x}}; \mathbf{X}\}$ is proportional to $\sqrt{\det(\mathbf{X})}$.*□

Proof Consider the singular value decomposition $\mathbf{X} = \mathbf{RDR}^\top$. Define $\{a, b, \cdots, z\}$ as being the ellipsoid's semi-axes and let $(a\,b \cdots z)$ be their product. From the fact that $\det(\mathbf{R}) = 1$ and from Proposition B.2 we have:

$$\det(\mathbf{X}) = \det(\mathbf{RDR}^\top) = \det(\mathbf{D}) = (a\,b \cdots z)^2.$$

The volume of an n-dimensional ellipsoid is

$$V_n \triangleq v_n\ (a\ b \cdots z) = v_n \sqrt{\det(\mathbf{X})},$$

where v_n is the volume of the n-dimensional unit sphere which is constant for a given dimension.[2] ■

B.4 Error propagation

Consider for instance the following non-linear function

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

of the independent Gaussian variables

$$\mathbf{x} \sim \mathcal{N}\{\bar{\mathbf{x}}, \mathbf{X}\} \qquad \mathbf{u} \sim \mathcal{N}\{\bar{\mathbf{u}}, \mathbf{U}\}.$$

At the output of the non-linear function, the variable $\mathbf{y}$ is no longer Gaussian. We aim to obtain the characteristics of a good Gaussian approximation. We perform the Taylor linearization of $\mathbf{f}$ (see (A.4) and (A.9)):

$$\mathbf{y} \approx \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) + \mathbf{F}_{\mathbf{x}}(\mathbf{x} - \bar{\mathbf{x}}) + \mathbf{F}_{\mathbf{u}}(\mathbf{u} - \bar{\mathbf{u}})$$

[2] For the lower dimensions we have $v_1 = 2$, $v_2 = \pi$ and $v_3 = \frac{4}{3}\pi$.

with the Jacobians

$$\mathbf{F}_\mathbf{x} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}^\top}\bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \qquad \mathbf{F}_\mathbf{u} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}^\top}\bigg|_{\bar{\mathbf{x}},\bar{\mathbf{u}}}$$

We apply the mean and covariances matrix definitions (B.2) and the linear properties of the expectation operator. We obtain:

$$\bar{\mathbf{y}} \approx \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \tag{B.6}$$

and:

$$\mathbf{Y} \approx \mathbf{F}_\mathbf{x}\mathbf{X}\mathbf{F}_\mathbf{x}^\top + \mathbf{F}_\mathbf{u}\mathbf{U}\mathbf{F}_\mathbf{u}^\top. \tag{B.7}$$

If $\mathbf{x}$ and $\mathbf{u}$ have cross-correlations $\mathbf{C}_{\mathbf{xu}} = \mathbf{C}_{\mathbf{ux}}^\top$, we can opt for considering them as a single Gaussian variable $\mathbf{z}$ characterized by

$$\mathbf{z} \triangleq \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \sim \mathcal{N}\left\{ \begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{u}} \end{bmatrix} ; \begin{bmatrix} \mathbf{X} & \mathbf{C}_{\mathbf{xu}} \\ \mathbf{C}_{\mathbf{ux}} & \mathbf{U} \end{bmatrix} \right\}.$$

Then

$$\bar{\mathbf{y}} \approx \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{u}}) \tag{B.8}$$

and

$$\mathbf{Y} \approx \begin{bmatrix} \mathbf{F}_\mathbf{x} & \mathbf{F}_\mathbf{u} \end{bmatrix} \begin{bmatrix} \mathbf{X} & \mathbf{C}_{\mathbf{xu}} \\ \mathbf{C}_{\mathbf{ux}} & \mathbf{U} \end{bmatrix} \begin{bmatrix} \mathbf{F}_\mathbf{x}^\top \\ \mathbf{F}_\mathbf{u}^\top \end{bmatrix}. \tag{B.9}$$

Appendix C

Miscellanies

C.1 Converting rotation representations

We give the formulae for the six possible conversions between the three rotation representations used in this thesis: the rotation matrix, the Euler angles and the quaternion.

Euler angles to rotation matrix

Given the Euler angles $\mathbf{e} = [\phi\ \theta\ \psi]^\top$ corresponding to the *roll*, *pitch* and *yaw* orientations in the ZYX convention, the corresponding rotation matrix is given by

$$\mathbf{R} = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{C.1}$$

We can name this function e2R$(\cdot)$ and write

$$\mathbf{R} = \mathrm{e2R}(\mathbf{e}).$$

Quaternion to rotation matrix

Given the quaternion $\mathbf{q} = [a\ b\ c\ d]^\top$ the rotation matrix corresponding to the same rotation is given by

$$\mathbf{R} = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2\cdot(bc-ad) & 2\cdot(bd+ac) \\ 2\cdot(bc+ad) & a^2-b^2+c^2-d^2 & 2\cdot(cd-ab) \\ 2\cdot(bd-ac) & 2\cdot(cd+ab) & a^2-b^2-c^2+d^2 \end{bmatrix}. \tag{C.2}$$

We can name this function q2R$(\cdot)$ and write

$$\mathbf{R} = \mathrm{q2R}(\mathbf{q}).$$

Rotation matrix to Euler angles

Given the rotation matrix

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{C.3}$$

the three Euler angles $\mathbf{e} = [\phi\ \theta\ \psi]^\top$ corresponding to the same rotation are given by

$$\begin{aligned} \phi &= \arctan(r_{32}/r_{33}) \\ \theta &= \arcsin(-r_{31}) \\ \psi &= \arctan(r_{21}/r_{11}) \end{aligned} \tag{C.4}$$

where the four quadrant version of the function $\arctan(\cdot)$ is used. We can name this function $\mathrm{R2e}(\cdot)$ and write

$$\mathbf{e} = \mathrm{R2e}(\mathbf{R}).$$

Rotation matrix to quaternion

Given the rotation matrix (C.3), the quaternion $\mathbf{q} = [a\ b\ c\ d]^\top$ corresponding to the same rotation is given by Algorithm C.1. At first sight it may seem a long algorithm, but it is simply a test for eventual singularities that results in the selection of a particular (one over four) small group of equations. We can name this function $\mathrm{R2q}(\cdot)$ and write

$$\mathbf{q} = \mathrm{R2q}(\mathbf{R}).$$

Euler angles to quaternion

This conversion is best achieved by using quaternion algebra. We first define the three quaternions corresponding to the three elementary Euler rotations:

$$\mathbf{q}_\phi = \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{q}_\theta = \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \qquad \mathbf{q}_\psi = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix}$$

The composed rotation multiplies them up as $\mathbf{q} = \mathbf{q}_\psi \cdot \mathbf{q}_\theta \cdot \mathbf{q}_\phi$. We obtain

$$\mathbf{q} = \begin{bmatrix} \cos(\psi/2)\,\cos(\theta/2)\,\cos(\phi/2) + \sin(\psi/2)\,\sin(\theta/2)\,\sin(\phi/2) \\ \cos(\psi/2)\,\cos(\theta/2)\,\sin(\phi/2) - \sin(\psi/2)\,\sin(\theta/2)\,\cos(\phi/2) \\ \cos(\psi/2)\,\sin(\theta/2)\,\cos(\phi/2) + \sin(\psi/2)\,\cos(\theta/2)\,\sin(\phi/2) \\ -\cos(\psi/2)\,\sin(\theta/2)\,\sin(\phi/2) + \sin(\psi/2)\,\cos(\theta/2)\,\cos(\phi/2) \end{bmatrix} \tag{C.5}$$

We can name this function $\mathrm{e2q}(\cdot)$ and write

$$\mathbf{q} = \mathrm{e2q}(\mathbf{e}).$$

Quaternion to Euler angles

Given the quaternion $\mathbf{q} = [a\ b\ c\ d]^\top$, the Euler angles $\mathbf{e} = [\phi\ \theta\ \psi]^\top$ corresponding to the same rotation are given by

$$\begin{aligned} \phi &= \arctan\left(\frac{2cd + 2ab}{a^2 - b^2 - c^2 + d^2}\right) \\ \theta &= \arcsin\left(-2bd + 2ac\right) \\ \psi &= \arctan\left(\frac{2bc + 2ad}{a^2 + b^2 - c^2 - d^2}\right) \end{aligned} \tag{C.6}$$

Algorithm C.1 Rotation matrix to quaternion conversion algorithm.

```
Require: $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$
  $T = 1 + trace(\mathbf{R})$
  if $(T > 10^{-8})$ then
    $S = 2\sqrt{T}$
    $a = S/4$
    $b = (r_{32} - r_{23})/S$
    $c = (r_{13} - r_{31})/S$
    $d = (r_{21} - r_{12})/S$
  else if $(r_{11} > r_{22})$ and $(r_{11} > r_{33})$ then
    $S = 2\sqrt{1 + r_{11} - r_{22} - r_{33}}$
    $a = (r_{23} - r_{32})/S$
    $b = -S/4$
    $c = (r_{21} - r_{12})/S$
    $d = (r_{13} - r_{31})/S$
  else if $(r_{22} > r_{33})$ then
    $S = 2\sqrt{1 - r_{11} + r_{22} - r_{33}}$
    $a = (r_{31} - r_{13})/S$
    $b = (r_{21} - r_{12})/S$
    $c = -S/4$
    $d = (r_{32} - r_{23})/S$
  else
    $S = 2\sqrt{1 - r_{11} - r_{22} + r_{33}}$
    $a = (r_{12} - r_{21})/S$
    $b = (r_{13} - r_{31})/S$
    $c = (r_{32} - r_{23})/S$
    $d = -S/4$
  end if
```

where the four quadrant version of the function arctan(·) is used. We can name this function q2e(·) and write

$$\mathbf{e} = \mathrm{q2e}(\mathbf{q}).$$

References

Abel, N. H. 1928. Sur la résolution algébrique des équations.

Agrawal, M., Konolige, K., and Iocchi, L. 2005. Real-time detection of independent motion using stereo. In *Proceedings of the IEEE Workshop on Motion and Video Computing (WACV/MOTION'05)*. Vol. 2. IEEE Computer Society, Washington, DC, USA, 207–214.

Alspach, D. L. and Sorenson, H. W. 1972. Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE transactions on automatic control 17,* 4, 439–448.

Bailey, T. 2003. Constrained initialisation for bearing-only SLAM. *IEEE International Conference on Robotics and Automation 2*, 1966– 1971.

Bailey, T. and Durrant-Whyte, H. 2006. Simultaneous localisation and mapping (SLAM): Part II state of the art.

Breakwell, J. V. 1967. Estimation with slight non-linearity. Unpublished communication.

Castellanos, J. A., Neira, J., and Tardós, J. D. 2004. Limits to the consistency of the EKF-based SLAM. In *5th IFAC Symposium on Intelligent Autonomous Vehicles*. Lisboa, PT.

Davison, A. 2003. Real-time simultaneous localisation and mapping with a single camera. In *Proc. International Conference on Computer Vision*. Nice.

Devy, M., Garric, V., and Orteu, J. 1997. Camera calibration from multiple views of a 2D object, using a global non linear minimization method. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'97)*. Grenoble, France.

Doucet, A., de Freitas, N., and Gordon, N. 2001. *Sequential Monte Carlo Methods in Practice.* Springer Verlag, New York.

Durrant-Whyte, H. and Bailey, T. 2006. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms.

Eade, E. and Drummond, T. 2006. Scalable monocular SLAM. *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition 1*, 469–476.

Estrada, C., Neira, J., and Tardós, J. 2005. Hierarchical SLAM: real-time accurate mapping of large environments. *IEEE Transactions on Robotics 21,* 4, 588 – 596.

Eustice, R., Singh, H., and Leonard, J. 2005. Exactly sparse delayed-state filters. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Barcelona, Spain, 2428–2435.

Farid, H. and Simoncelli, E. P. 1997. Optimally rotation-equivariant directional derivative kernels. In *Proceedings of Computer Analysis of Images and Patterns*. Kiel, Germany.

Faugeras, O. 1993. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* The MIT Press.

Fischler, M. and Bolles, R. 1981. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. of ACM. 24(6)*, 381–395.

Foxlin, E. M. 2002. Generalized architecture for simultaneous localization, auto-calibration, and map-building. In *IEEE/RSJ Conf. on Intelligent Robots and Systems.*

Grewal, S. and Andrews, A. P. 1993. *Kalman Filtering, Theory and Practice.* Prentice Hall, Englewood Cliffs, New Jersey.

Harris, C. and Stephens, M. 1988. A combined corner and edge detector. In *Fourth Alvey Vision Conference*. Manchester (UK).

Hartley, R. and Zisserman, A. 2000. *Multiple View Geometry in Computer Vision.* Cambridge University Press.

Horaud, R. P. and Monga, O. 1995. *Vision par ordinateur: outils fondamentaux.* Éditions Hermès, Paris.

Jazwinski, A. H. 1970. *Stochastic Processes and Filtering Theory*. Academic Press, NY.

Kalman, R. 1960. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 35–45.

Konolige, K. 2005. SLAM via variable reduction from constraints maps. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. Barcelona, 667–672.

Kronham, T. R. 1998. Bearing only target motion analysis based on a multihypothesis Kalman filter and adaptative ownship motion control. In *IEEE Proceedings on Radar Sonar and Navigation*. Vol. 145. 247–252.

Kullback, S. 1959. *Information theory and statistics*. John Wiley and Sons., New York.

Kwok, N. M. and Dissanayake, G. 2004. An efficient multiple hypothesis filter for bearing-only SLAM. In *IEEE/SRJ International Conference on Intelligent Robots and Systems*. Sendai, Japan.

Kwok, N. M., Dissanayake, G., and Ha, Q. P. 2005. Bearing-only SLAM using a SPRT based Gaussian sum filter. In *Proc. IEEE Int. Conf. on Robotics and Automation*. Barcelona.

Lacroix, S., Devy, M., Solà, J., and Lemaire, T. 2005. Modélisation 3D par vision pour la robotique mobile : approches de cartographie et localisation simultanées. *Revue Française de Photogrammétrie et de Télédétection* 180, 26–40.

Le Cadre, J. and Jauffret, C. 1997. Discrete-time observability and estimability analysis for bearings-only target motion analysis. *IEEE Transactions on Aerospace and Electronic Systems 33,* 1 (January), 178–201.

Lemaire, T., Lacroix, S., and Solà, J. 2005a. Experiments with a bearing-only SLAM algorithm. Tech. Rep. 05201, LAAS-CNRS.

Lemaire, T., Lacroix, S., and Solà, J. 2005b. A practical 3D bearing only SLAM algorithm. In *IEEE International Conference on Intelligent Robots and Systems*. Edmonton,Canada.

Lisien, B., Morales, D., Silver, D., Kantor, G., Rekleitis, I., and Choset, H. 2003. Hierarchical simultaneous localization and mapping. In *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*. Vol. 1. 448–453.

Lourakis, M. and Argyros, A. 2004. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece. Aug. Available from `http://www.ics.forth.gr/~lourakis/sba`.

Lowe, D. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 91–110.

Ma, Y., Soatto, S., secká, J. K., and Sastry, S. S. 2004. *An Invitation to 3-D Vision*. Springer-Verlag, New York.

Mallet, A., Lacroix, S., and Gallo, L. 2000. Position estimation in outdoor environments using pixel tracking and stereovision. In *International Conference on Robotics and Automation*. San Francisco, CA (USA).

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. 2003. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI, Acapulco, Mexico.

Montiel, J., Civera, J., and Davison, A. 2006. Unified inverse depth parametrization for monocular SLAM. In *Proceedings of Robotics: Science and Systems*. Philadelphia, USA.

Newman, P. 1999. On the structure and solution of the simultaneous localisation and map building problem. Ph.D. thesis, Australian Centre for Field Robotics - The University of Sydney.

Noble, A. 1989. Descriptions of image surfaces. Ph.D. thesis, Department of Engineering Science, Oxford University.

Peach, N. 1995. Bearing-only tracking using a set of range-parametrised extended kalman filters. In *IEEE Proceedings on Control Theory Applications*. 73–80.

Shi, J. and Tomasi, C. 1994. Good features to track. In *Proc. IEEE Int. Conf. on Computer Vision and Pattern Recognition*. Seattle, USA, 593–600.

Smith, R. and Cheeseman, P. 1987. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research 5,* 4, 56–68.

Solà, J. 2005. Delayed vs undelayed landmark initialization for bearing only SLAM. Extended abstract for the SLAM Workshop of the IEEE Int. Conf. on Robotics and Automation. Available from http://webdiis.unizar.es/~jdtardos/ICRA2005/SolaAbstract.pdf.

Solà, J., Monin, A., and Devy, M. 2006. Incremental structure from restrictive motion. Tech. Rep. 06597, LAAS-CNRS, Toulouse, France. Available from http://www.laas.fr/~jsola/publications/solaBMVC06.pdf.

Solà, J., Monin, A., and Devy, M. 2007. BiCamSLAM: Two times mono is more than stereo. In *International Conference on Robotics and Automation*. Rome, Italy. Accepted for publication. Available from http://www.laas.fr/~jsola/publications/solaICRA2007.pdf.

Solà, J., Monin, A., Devy, M., and Lemaire, T. 2005. Undelayed initialization in bearing only SLAM. In *IEEE International Conference on Intelligent Robots and Systems*. Edmonton, Canada. Available from http://www.laas.fr/~jsola/publications/UndelayedBOSLAM.pdf.

Strobl, K., Sepp, W., Fuchs, S., Paredes, C., and Arbter, K. 2006. Camera calibration toolbox for Matlab. Tech. rep., Institute of Robotics and Mechatronics, Wessling, Germany. Available from http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

Thrun, S., Liu, Y., Koller, D., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H. 2004. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research 23,* 7-8, 693–716.

Tupysev, V. A. 1998. A generalized approach to the problem of distributed Kalman filtering. In *AIAA Guidance, Navigation and Control Conference*. Boston.

Vidal, R., Ma, Y., Soatto, S., and Sastry, S. 2002a. Segmentation of dynamic scenes from the multibody fundamental matrix. In *Proc. of the ECCV workshop on Vision and Modeling of Dynamic Scenes*.

Vidal, R., Ma, Y., Soatto, S., and Sastry, S. 2002b. Two-view segmentation of dynamic scenes from the multibody fundamental matrix. Tech. Rep. UCB/ERL M02/11, UC Berkeley. May.

Wang, C.-C. 2004. Simultaneous localization, mapping and moving object tracking. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Zhang, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc. of the Int. Conf. on Computer Vision*. Corfu, Greece, 666–673.

"Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach"

Abstract:
In this thesis we give new means for a machine to understand complex and dynamic visual scenes in real time. In particular, we solve the problem of simultaneously reconstructing a certain representation of the world's geometry, the observer's trajectory, and the moving objects' structures and trajectories, with the aid of vision exteroceptive sensors. We proceeded by dividing the problem into three main steps: First, we give a solution to the Simultaneous Localization And Mapping problem (SLAM) for monocular vision that is able to adequately perform in the most ill-conditioned situations: those where the observer approaches the scene in straight line. Second, we incorporate full 3D instantaneous observability by duplicating vision hardware with monocular algorithms. This permits us to avoid some of the inherent drawbacks of classic stereo systems, notably their limited range of 3D observability and the necessity of frequent mechanical calibration. Third, we add detection and tracking of moving objects by making use of this full 3D observability, whose necessity we judge almost inevitable. We choose a sparse, punctual representation of both the world and the moving objects in order to alleviate the computational payload of the image processing algorithms, which are required to extract the necessary geometrical information out of the images. This alleviation is additionally supported by active feature detection and search mechanisms which focus the attention to those image regions with the highest interest. This focusing is achieved by an extensive exploitation of the current knowledge available on the system (all the mapped information), something that we finally highlight to be the ultimate key to success.

Keywords: Vision, SLAM, dynamic scenes, real-time, undelayed initialization

"Vers la Cartographie et la Localisation Visuelles par un robot mobile, avec détection et suivi d'objets mobiles. Une approche Géométrique et Probabiliste"

Résumé :
Dans cette thèse, nous résolvons le problème de reconstruire simultanément une représentation de la géométrie du monde, de la trajectoire de l'observateur, et de la trajectoire des objets mobiles, à l'aide de la vision. Nous divisons le problème en trois étapes : D'abord, nous donnons une solution au problème de la cartographie et localisation simultanées pour la vision monoculaire qui fonctionne dans les situations les moins bien conditionnées géométriquement. Ensuite, nous incorporons l'observabilité 3D instantanée en dupliquant le matériel de vision avec traitement monoculaire. Ceci élimine les inconvénients inhérents aux systèmes stéréo classiques. Nous ajoutons enfin la détection et suivi des objets mobiles proches en nous servant de cette observabilité 3D. Nous choisissons une représentation éparse et ponctuelle du monde et ses objets. La charge calculatoire des algorithmes de perception est allégée en focalisant activement l'attention aux régions de l'image avec plus d'intérêt.

Mots Clefs : Vision, SLAM, objets dynamiques, temps-reel, initialsation immediate.