UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXVII Ciclo

Analysis and development of self localization and autonomous driving systems for the industrial and automotive environment.

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Massimo Bertozzi

Dottorando: *Nicola Bernini*

January 2015

To my wife and my love Chiara, Eddy and all our "dear friends"
To Maria, Sergio and Cristina
To all our friends, in Italy, Europe, China, Japan, . . .

Sommario

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduction

1.1 General Thesis Scope

The present manuscript regards the presentation of the main actitvities related to the author's doctoral experience.

The present work is aimed at studying the *self localization* problem, which is currently an active research topic, focusing on the development of milestones as working prototypes.

In this thesis, a multidisciplinary approach has been adopted and the problems are treated both from a theoritical and applicative perspective.

1.2 Thesis goals

The goal of the present thesis is dual. First the self localization problem is investigated in the industrial environment and a working prototype, consisting in an AGV able to perform self localization and autonomous driving under certain conditions, has been developed. Then the self localization problem is considered as a part of wider and currently very active research problem: the Simultaneous Localization And Mapping (SLAM), considered in the automotive environment.

The industrial application has been developed in collaboration with a local company which provided some of the necessary infrastructures. At the end of the presented milestone, the company feedback has been positive nevertheless it has been forced to temporarily put on hold its support for further development.

The plan for the automotive application development requires only the Vislab support.

Regarding the automotive environment, the present thesis goal comprises approaching the SLAM problem. In particular a mapping application has been developed, using a widely known automotive dataset, and localization is performed exploiting this map.

A roadmap to realize a complete SLAM system to be applied on Vislab autonomous vehicles has been defined.

Capitolo 2

Preliminaries

In this section a brief literature overview regarding the main topics treated in this thesis is provided.

The theory regarding *Bayesian Filtering* has been used to approach all of the localization problems presented in this work.

The *Inverse Kinematics* theory is briefly outlined because it is used to control the AGV.

Approaching the *Simultaneous Localization and Mapping (SLAM)* problem represents the main goal of the second part of this thesis hence an overview about the main concepts is provided. Also *Markov Random Fields (MRF)* are introduced because of a possible application in the SLAM context.

The *Bundle Adjustment* theory is presented because it is at the core of the *Structure from Motion* problem treated in the second part of the work while dealing with the map building process.

Solving the *location recognition* problem in an effective and efficient way is necessary to achieve a good SLAM implementation, hence the *visual vocabulary tree* theory is presented.

2.1 Bayesian Filtering

2.1.1 Particle Filter

Let's consider a system evolving over time according to some evolution law.

Let's assume a model that approximates the evolution law exists in the form of an *evolution function* (or more generally an evolution operator) acting on the *system state* which is assumed to be an element of a certain *state space*.

$$x(t) = f(\{x(\tau)\}_{\tau<t}, \{v_\tau\}_{\tau<t}) + n_{ev}(t) \tag{2.1}$$

with

- Ω state space
- $\mathcal{V}$ forcing term space
- $x(t) \in \Omega$ the system state at a certain time
- $\{x(\tau)\}_{\tau<t}$ history of the system evolution or equivalently trajectory in the system state space
- $v(t)$ forcing term at a certain time
- $\{v(\tau)\}_{\tau<t}$ history of the forcing term
- $f : \Omega \times \mathcal{V} \to \Omega$ Evolution Function / Evolution Operator
- $n_{ev}(t)$ IID random variable extracted from the evolutionary noise stochastic process

In the vast majority of physical systems models the *evolution* is represented by means of the Hamiltonian Function / Hamiltonian Operator which is included in conservation equation.

For example, Quantum Mechanical Systems evolution is defined by the Schroedinger's Equation which in turn depends on the Hamiltonian Operator.

Let's also assume noisy measurements are available at some moments.

$$z(t) = h(x(t)) + n_{obs}(t) \tag{2.2}$$

with

- $\mathscr{Z}$ observation space
- $z(t) \in \mathscr{Z}$ observation at a certain time
- $h : \Omega \to \mathscr{Z}$ Observation Function
- $n_{obs}(t)$ observation noise

Both the evolution function and the noisy measurements provide a partial approximating representation of the system state hence following a Data Fusion approach could lead to a better approximation.

Essentially it is possible to consider the evolving system state like an unknown stochastic process and the *state tracking* is the goal. An approximating (stochastic) differential equation for the system evolution is available as well as noisy measurements.

Considering this probabilistic paradigm, let's switch to a probabilistic view of the problem introducing the $P(x(t))$ *System State Probability Density Function* (System State PDF).

Let's observe that while $x(t)$ represents the real system state at a certain time which is unknown by definition, the $P(x(t))$ represents all the known information about the system state at a certain time.

It is possible to define $\hat{x}(t)$ *best state estimate* from the system state PDF for example in terms of average or mode.

Obviously the closer $\hat{x}(t)$ is to $x(t)$ the better the estimate.

In this context, the above described f *evolution function* becomes

$$P(x(t)|\{x_\tau\}_{\tau<t}, \{v(\tau)\}_{\tau<t})$$

the *probabilistic evolution functional*.

An implicit assumption related to typical implementations regards the stochastic process representing the evolving system state to respect the Markov Property: the future state depends just on the present state, no long memory effects. This transforms (2.1) in (2.3) as follows

$$\dot{x}(t) = f(x(t), v(t)) + n_{ev}(t) \tag{2.3}$$

moreover the *evolution functional* gets transformed as follows $P(\dot{x}(t)|x(t), v(t))$.

When considering particle filter in an applicative context it is common to quit the continuous time formalism switching to a discrete time one to make the notation more clear, because of the context it is possible to do this without loss of generality.

Under the Markov Property assumption and with the discrete time formalism, it is possible to compute the *evolution functional* to evolve the system until a certain time, just sequentially composing the atomic *evolution functionals* as described by the Chapman-Kolmogorov equation as follows

$$P(x_n|P_s) = \int_\Omega P(x_n|x_r)P(x_r|x_s)dx_r \qquad s < r < n \tag{2.4}$$

Hence to compute the evolution of a single step it follows

$$P(x_k) = \int P(x_k|x_{k-1})dP(x_{k-1}) = \int_\Omega P(x_k|x_{k-1})P(x_{k-1})dx_{k-1} \tag{2.5}$$

The Particle Filter is a recursive Bayesian filter that can be used to perform Data Fusion. Essentially it is a Monte Carlo strategy to sample the unknown System State PDF generated by a generic (also non linear) evolution functions and a generic (also non gaussian) noise. Like all the Monte Carlo methods, the precision can be improved increasing the number of samples but it requires higher computational power: essentially Monte Carlo methods impose a trade off between precision and computational cost.

It is a Bayesian Filter because it fuses, in a Bayesian fashion, two above defined sources of information: the evolution model and the noisy measurements.

The particle filter iteratively processes the data using the last iteration output as the input for the new iteration. Each iteration is divided in two subphases: Predict and Update.

In the Bayesian Framework context the above presented formalism is slightly modified to incorporate the z_k measurement. Two kinds of PDF exist:

- $P(x_k|z_{1:k-1})$ *Evolved Prior PDF* that incorporates all the information up to a certain time excepting the last measure
- $P(x_k|z_{1:k})$ *Posterior PDF* that incorporates all the information up to a certain time

In the **Predict Subphase** the previously computed Posterior System State PDF is evolved according to the approximating evolution model.

$$P(x_k|z_{1:k-1}) = \int P(x_k|x_{k-1})P(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.6}$$

with

- $P(x_k|z_{1:k-1})$ *Evolved Prior PDF*
- $P(x_k|x_{k-1})$ Atomic Transition Functional
- $P(x_{k-1}|z_{1:k-1})$ Posterior PDF computed at the previous iteration

In the particle filter context, PDFs have no known close form, they are represented by a set of samples hence the evolution function is applied to each sample.

In the ***Update Subphase*** the Evoluted PDF is fused with the result of the measurement, using the Likelihood Function which expresses the compatibility between the evolved state PDF and the measurement itself.

$$P(x_k|z_{1:k}) = \frac{P(z_k|x_k)P(x_k|z_{1:k-1})}{P(z_k|z_{1:k-1})} \tag{2.7}$$

with

- $P(x_k|z_{1:k})$ *Posterior PDF*

- $P(z_k|x_k)$ Likelihood Function
- $P(x_k|z_{1:k-1})$ Evolved Prior PDF
- $P(z_k|z_{1:k-1})$ Normalization Constant

Essentially the normalization constant is computed imposing the normalization of the numerator

$$P(z_k|z_{1:k-1}) = \int_{\Omega} P(z_k|x_k)P(x_k|z_{1:k-1})dx_k \tag{2.8}$$

Again under the particle filter framework, the compatibility is not measured considering the posterior PDF, which is unknown, but considering the samples or equivalently the particle set.

In figure 2.1 an example of particle filter for indoor localization is shown.

Each red point is a *particle* namely a possible system state.

The whole particle set represents a sampling of the unknown *System State PDF* hence the higher the number of particles the more precise the sampling but this improvement comes at the cost of a higher computational power need.

Particles are evolved according to the internal system model.

When available, Measurements Data are fused with the evolved system PDF by means of the Likelihood Function as previously described.

In general, the system free evolution make the system PDF spread while the measurements make the system PDF collapse close to one or more states, similarly as it happens for Quantum Mechanics systems.

2.1.2 Kalman Filter family

The **Kalman Filter** [10] descends from the Particle Filter equations assuming the evolution function is linear and the noise is Gaussian. Under these assumptions, the Kalman Filter is optimal.

The assumptions regarding the Kalman Filter are quite strong for real world applications hence two other variants with relaxed assumptions have been developed

- the Extended Kalman Filter

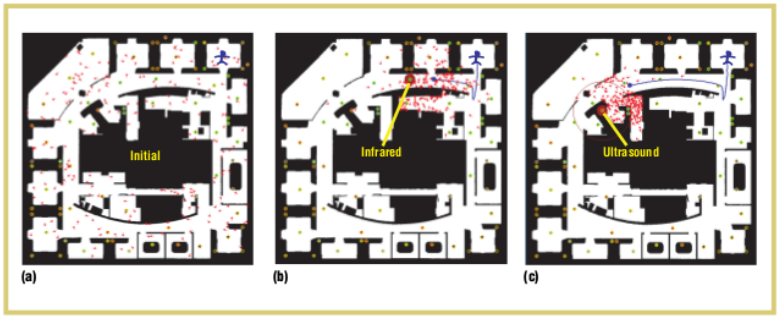


Figura 2.1: Example of indoor particle filter. Fox et al. [1].

- the Unscented Kalman Filter

2.2 Inverse Kinematics Problem

The *Inverse Kinematics Problem* for mobile control is well studied and a wide literature exists about the control of mobile robots even with a complex dynamics.

An interesting approach is the one provided by Gracia et al. [11] that regarded a fork lift as for the prototype presented in the present work.

In this thesis a very simplice approach has been adopted and the satisfying quality of the final results justified this choice in this early development stage.

The *kinematics model* that has been adopted to approximate the fork lift one is the simple *tricycle model* with one steering wheel and two motor wheels.

A fundamental parameter in this model is L representing the distance between the steering wheel and the middle point on the axis of the motor wheels.

Considering the tricycle can only move on a plane depending on $\phi(t)$ *steering wheel angle* at a certain time, then assuming constant speed, the goal is to find such a function according to some desiderable trajectory for the vehicle.

In this thesis, the trajectory is determined computing two cubic splines $x(t), y(t)$ related to the motion on the two orthogonal axis.

The driving function can be computed according to the following known formula for the above mentioned vehicle model

$$k(t) = \frac{(\ddot{x}(t)\dot{y}(t)) - (\dot{x}(t)\ddot{y}(t))}{(\dot{x}(t)^2 + \dot{y}(t)^2)^{3/2}} \tag{2.9}$$

with

- $k = \frac{1}{R}$ the inverse of the curvature radius

Finally the *steering wheel function* can be computed as follows

$$\phi(t) = \arctan(Lk(t)) \tag{2.10}$$

2.3 SLAM

The Simultaneous Localization and Mapping (SLAM) is a problem in the context of mobile robotics that has recently attracted the attention of many researchers.

Given a moving robot in an unknown environment, essentially it consists in the estimation of the *robot pose* and *map* at the same time.

It is the fusion of two problems

- Localization: estimating the robot position given a map
- Mapping: estimating the map given robot locations

It is a complex problem because the Localization Problem depends on the Map and the Mapping Problem depends on the Localization.

The mapping estimation problem and the localization estimation problem are processes affected by errors, in the SLAM context the errors related to each process correlate and propagate in both the processes.

In general a SLAM Problem consists of elaborating a set of robot odometrical data and of relative observations (i.e. computer vision data) to compute a map and the path of the robot in the map.

Essentially the complexity of the problem lies in transforming the *relative information* in *absolute information* hence moving from *relative landmark positions* and *relative robot poses* to *absolute and coherent landmark positions* and *absolute and coherent robot poses*.

Additional complexity could come from time constraints related with the problem: Offline SLAM, that consists of the live recording ofall the odometrical data and the relative observations for later trying to solve the SLAM problem, is inherently easier than Online SLAM, that consists of trying to solve the same problem live, while moving in the environment.

2.3.1 SLAM - Formalism

Let's briefly introduce the SLAM Formalism commonly used in literature so to better introduce the key concepts.

First let's assume a discrete time context then let's introduce the following concepts

- $x(t)$ *vehicle state vector* containing the information about the location and orientation. Let's observe two kind of notations are possible
  - $x(t) \in \mathbb{R}^6$ minimal vector notation
  - $x(t) \in SE(3)$ rototranslation matrix notation

  and that it is possible to switch from one notation to the other without any problem.
- $u(t)$ *control vector* or *wheel odomerty* information
- $m_i \in \mathbb{R}^3$ *3D coordinates* of a certain *landmark*. Landmark position is assumed to be invariant over time.
- *m 3D coordinates of all the landmarks* hence the *map*

- $z_i(t)$ *observation of certain landmark taken from the vehicle at a certain time*
- $z(t)$ *set of all the landmarks observed at a certain time*
- $\{x(\tau)\}_{\tau=1:t}$ History of vehicle locations and orientations
- $\{u(\tau)\}_{\tau=1:t}$ History of vehicle controls or wheel odometry
- $\{z(\tau)\}_{\tau=1:t}$ History of all the landmarks observations

The SLAM problem is represented following to a **probabilistic paradigm** hence the goal is computing for every time instant the following distribution

$$P(x(t), m|\{u_\tau\}, \{z(\tau)\}, x_0) \tag{2.11}$$

which is the *Joint Posterior PDF* regarding the vehicle position (solution of the localization problem) and the map (solution of the mapping problem).

The single **mapping problem** essentially consists of the computation of

$$P(m|\{u_\tau\}, \{z_\tau\}, \{x(\tau)\}) \tag{2.12}$$

hence a complete knowledge of $\{x(\tau)\}$ *position history* is assumed.

The single **localization problem** essentially consists of the computation of

$$P(x(t)|\{u(\tau)\}, \{z(\tau)\}, m) \tag{2.13}$$

hence a complete knowledge of *m map* is assumed.

The complexity of the SLAM problem make it impossible to factorize the Joint PDF as follows

$$P(x(t), m) \neq P(x(t))P(m) \tag{2.14}$$

because of the mutual correlations hence it is necessary to perform an estimation of both the elements at the same time.

2.3.2 SLAM - Recursive Bayesian Filtering

It is possible to approach the problem of estimating the SLAM joint posterior PDF following a *Recursive Bayesian Filtering* approach.

This strategy consists of defining a *predict model* and an *update model* and to iteratively apply them from an initial known PDF until all the known information has been integrated.

The **predict model** involves no $z(t)$ *observation* hence it concerns only the *vehicle position estimation* using the *vehicle model* hence it is as follows

$$P(x(t)|x(t-1),u(t)) \tag{2.15}$$

The *evolved prior* then becomes

$$P(x(t),m|\{z(1:t-1)\},\{u(1:t)\},x_0) = \int P(x(t)|x(t-1),u(t))\cdot \\ P(x(t-1),m|\{z(1:t-1)\},\{u(1:t-1)\},x_0)dx(t-1) \tag{2.16}$$

The **update model** involves the $z(t)$ *observation* tp update the prediction hence a sort of *likelihood function* is needed as follows

$$P(z(t)|x(t),m) \tag{2.17}$$

With this function defined it is possible to apply the Bayes' rule to compute the *Joint Posterior PDF* depending on the *Evolved Prior PDF* previously defined and hence depending on the Joint Posterior estimated at the end of the previous iteration

$$P(x(t),m|\{z(1:t)\},\{u(1:t)\},x_0) = \frac{P(z(t)|x(t),m)P(x(t),m|\{z(1:t-1)\},\{u(1:t)\},x_0)}{P(z(t)|\{z(1:t-1)\},\{u(1:t)\})} \tag{2.18}$$

The classical solution to SLAM is the one provided by Smith and Cheeseman [12] that used an EKF as Recursive Bayesian Filter to estimate the joint distribution of the robot pose and landmark position. The EKF requires to assume Gaussian Noise and as a result the joint PDF is constrained to be Gaussian Distribution. Moreover it has been observed that the *covariance matrix update* scales as $O(n^2)$ with n depending on the map dimensions hence this method does not scale well to large maps.

The SLAM is known to pose scalability problem also regarding other tasks, like the *loop closure detection* that is essentially equivalent to an image search problem over large datasets.

2.3.3 SLAM - Dynamic Bayesian Networks

A typical representation formalism for the SLAM problem is the **Dynamic Bayesian Network** as in Figure 2.2.

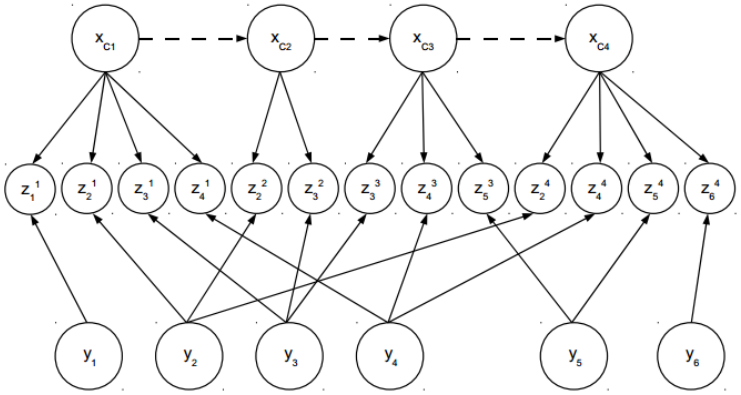


Figura 2.2: SLAM Dynamic Bayesian Network representation

The initial state represented in a white square is known, while the blue circles represent the *observed variables* and the white circles the *hidden variables* that need to be estimated.

Each variable in a circle, being a random variable, is represented as PDF. For example the $\{u_i\}$ commands (or wheel odometrical data) are not known with absolute certainty but usually they are represented as a Gaussian Distribution with (μ, σ) known information.

Considering the resulting graph is a chain, this formalism underlines the *temporal structure* of the problem that can effectively be treated by means of the above

described Recursive Bayesian Filter Framework. Essentially two main strategies are possible depending on the assumptions:

- assuming Gaussian noise, it is possible to use the Extended Kalman Filter (EKF)
- relaxing all the assumptions regarding the linearity of the predict model and the gaussianity of the noise, it is possible to use the Particle Filter

It is important to consider that the $P(z(t)|x(t),m)$ *observation model* is in general multimodal, hence the Gaussianity Assumption about the PDFs involved in the computation does not hold.

2.3.4 SLAM - Mapping: Metric Mapping and Topological Mapping

In the SLAM context, the *map learning* task needs a detailed analysis.

Essentially a *map* represents the result of a fusion between any *prior information* and the *observed information* about the environment.

It is possible to represent this information using *different levels of abstraction*.

The lowest level of abstraction possible is represented by the *sensors raw data* which constitutes a *metric map*.

Feature detection is a form of *pattern recognition* that summerizes the raw data world representation introducing a certain level of symbology hence producing a *metric feature based map*.

Slightly improving the level of abstraction, the *grid based maps* are found.

Essentially they use a grid-based world representation and each *cell* in the grid abstracts its real world content. This kind of approach is similar to some obstracle detection ones that build a grid like world representation consisting of 3 possible values

- free
- occupied
- unknown

The next abstraction layer is obtained by *segmenting* the metric map in *key places* and performing *grouping of visible landmarks* so to build a sort of *place fingerprint*. This kind of map is called a *topological map* because it is focused on representing the *environment topology*.

Essentially the *topological map* is equivalent to a *graph* where

- nodes represent *key places* with an associated *fingerprint* derived from *detected landmarks*
- edges represent *spatial vincula* among places

Getting to higher levels of abstraction is a fundamental task to be able to build maps which show *inavariance properties*. Good invariance properties allow for identifying the deep nature of an element, which is invariant with respect to some transformations. Essentially the most desiderable property of the above mentioned *place fingerprint* should be an invariance to common environment transformations over time.

2.3.5 SLAM - Pose Graph

The localization analogous concept for the mapping concept of the topological map is the *pose graph*.

Essentially a *Pose Graph* is a $G = (G_V, G_E)$ graph where

- G_V is the set of nodes
- G_E is the set of edges
- each $p_i \in G_V$ node represents a *pose*
- each $\delta_{i,j} \in G_E$ edge represents a *transformation* between the two poses

Considering p_i, p_j two poses hence two nodes on the graph, the edge $\delta_{i,j}$ representing the transformation between these two nodes can be defined as from the following equation

$$p_j = p_i \oplus \delta_{i,j} \tag{2.19}$$

In this framework, let's also define z_i as the measurement related to the p_i node hence $\delta_{i,j}^{(z)}$ is the *transformation* that make z_i maximally overlap z_j as follows

$$\delta_{i,j}^{(z)} = \arg\min_{\delta_{i,j}^{(z)}} d_{ol}(\delta_{i,j}^{(z)}(z_i), z_j) \tag{2.20}$$

considering $\mathscr{Z}$ the Observation Space, with

$$d_{ol} : \mathscr{Z} \times \mathscr{Z} \to \mathbb{R}^+$$

an overlapping measure for the observations.

Moreover knowing $\delta_{i,j}$ (inverting 2.19) it is possible to apply the *Observation Operator* to both sides of the equation computing $\hat{\delta}_{i,j}^{(z)}(p_i, p_j)$ *predicted transformation* which depends on the two nodes poses.

Thus for $G'_E \subset G_E$ subset of the edges for which measurement information is available, the following elements can be computed

- $\delta_{i,j}^{(z)}$ the observed maximum overlapping transformation
- $\hat{\delta}_{i,j}^{(z)}$ the predicted transformation

then it is possible to define an error function as follows

$$e_{i,j}(p_i, p_j) = \delta_{i,j}^{(z)} - \hat{\delta}_{i,j}^{(z)}(p_i, p_j) \tag{2.21}$$

in this formulation, it has been pointed out as this function depends on the two poses values.

Finally a pairwise Maximum Likelihood Estimation problem can be set up defining the negative log likelihood as follows

$$F(p_j; p_i) = e_{i,j}(p_i, p_j)^T \Omega_{i,j} e_{i,j}(p_i, p_j) \tag{2.22}$$

where $\Omega_{i,j}$ is the *information matrix* computed by the measurements.

Let's observe that in this pairwise problem p_j is the variable while p_i has been considered as a parameter.

The problem can then be generalized considering the subgraph $G' = (G'_V, G'_E) \subset G$ for which measures are available, defining a global negative likelihood function as follows

$$F_{\{p_i\}_{i \in G_V}} = \sum_{\langle i,j \rangle \in G'_E} F(p_i, p_j) \tag{2.23}$$

hence the best global configuration satisfies

$$\{\hat{p}_i\}_{i \in G'_V} = \arg\min_{\{p_i\}} F_{\{p_i\}} \tag{2.24}$$

2.3.6 SLAM - Loop Closure Detection

Essentially the Loop Closure Detection is a problem of *location recognition* which is presented in section 2.8.

In the context of the previous analysis, it is straight to observe that a very important element to define the pose graph minimization problem is indeed the *loop closure detection* because it is the process that deeply transforms the pose graph topology, imposing further consistency vincula.

Without these kind of vincula, the dead recoking error will accumulate indefinitely hence continuously decreasing the quality of the self localization and mapping.

The loop closure detection requires a *prior map knowledge* because essentially it consists of the capability to recognize already visited locations.

Place recognition is a complex problem because this process should be tolerant to *environment variations*.

Recognizing a place already visited, not after a small time frame (hence assuming little modification in the images) like it happens for *stereo from motion* or *visual odometry*, but after longer time, hence taking into consideration important image variations (illumination conditions, little changes in the scene, ...) is a non trivial problem.

The nature of the problem lies in searching for *invariances* as it has been already pointed out in 2.3.4.

False negatives (not recognizing an already visited place) are in general less dangerous than false positives (recognizing a new a place as an already visited one)

because imposing even little wrong consistency vincula on the map could possibly quickly deteriorate it.

The problem of loop closure detection is inherently associated with the *location recognition* problem presented in section 2.8 which is essentially constituted of two other problems

- building a sort of *fingerprint* to be associated with each place that is as unique as possible while it is as invariant as possible to typical environmental changes (illuminations, ...)
- effectively and efficiently searching across large datasets (typical for real world SLAM applications)

2.4 SLAM as MRF

The work presented by Strasdat et al. [13] proposed an interesting approach to represent the online SLAM/SfM problem using the *Bayesian Network* formalism. Dynamic Bayesian Networks and Markov Random Fields are used to formalize the *sequential online dual estimation problem*. Nodes are random variables while edges represent statistical dependency relations among them.

In figure 2.3 a *Dynamic Bayesian Network* (DBN) representing the online SLAM problem is shown. The DBNs are directed and acyclic graphs.

It is composed of the following elements

1. The $\{X_{C_k}\}$ represents a sequence of camera positions
2. The $\{Y_i\}$ represents a sequence of landmarks positions
3. The $\{Z_i^k\}$ represents the observation related to a certin camera and a certain observed landmark

In the SLAM context, the *sequence of camera positions* 1 and the *sequence of landmark positions* 2 are sequences of random variables and their simultaneous estimation is the goal of the SLAM problem.

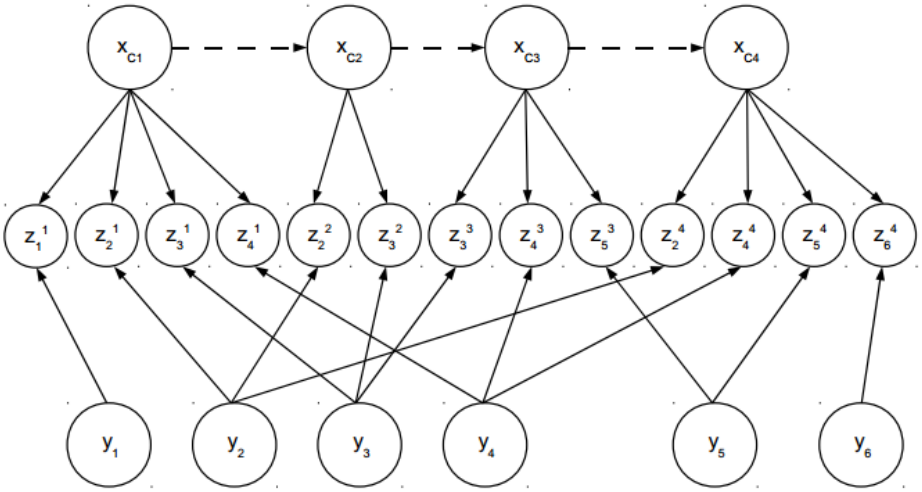


Figura 2.3: *SLAM formalized as a DBN.*

The *sequence of observations* 2 are the results of the measurements (Images) performed during the motion.

If the complete $\{X_{C_k}\}$ *poses sequence* is known (then it is not a random variable anymore) then it becomes an offline Structure from Motion problem. Essentially it consists of solving a global optimization regarding a Bundle Adjustment problem.

In the online SLAM problem, the *poses sequence* is not known but needs to be estimated according to odometrical data as well as to landmarks observation.

A common Bayesian formalism that is used to represent this kind of problem is the *Markov Random Field*. In figure 2.4 a MRF representing the online SLAM problem is shown. It is an analogous representation of the above mentioned DBN with the difference that MRFs are undirected graph that may be cyclical.

In an online SLAM problem the MRF grows in terms of $\{X_{C_k}\}$ *poses sequence* and may also grow in terms of $\{Y_i\}$ *observed landmarks*. Figure 2.5 outlines this concept.

To solve the online SLAM problem two main approaches are proposed

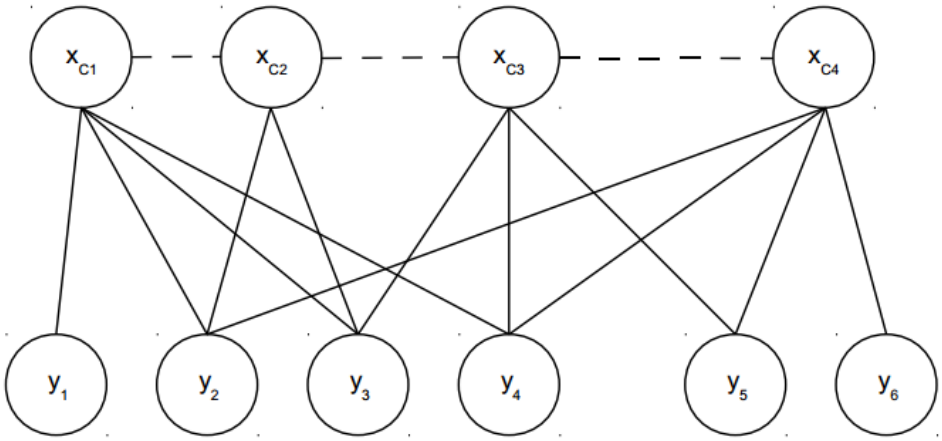


Figura 2.4: *SLAM formalized as a MRF.*

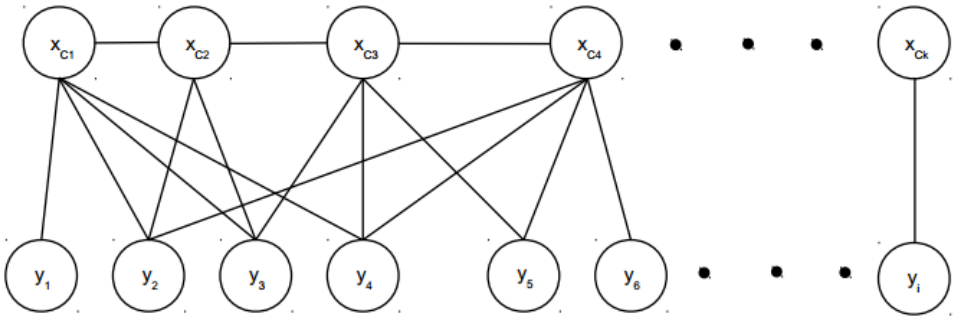


Figura 2.5: *SLAM formalized as a MRF.*

1. Keyframe based SLAM
2. Filtering based SLAM

The *Keyframe based SLAM* 1 consists of performing the following simultaneous operations

1. Assuming the map, performing localization at every step

2. Assuming a *subset of positions*, perform mapping

The *localization task* 1 can be performed with a typical Bayesian Filter to fuse the information about the *prior*, derived by the odometrical data, and the *measurements*, derived by the landmarks observations, to compute the *posterior* of the position.

The *mapping task* 2 requires identifying a set of *keyframes* namely a $\{X_{C_{k'}}\} \subset \{X_{C_k}\}$ *subset of camera poses* accordng to some criteria, to be assumed so to carry out Bundle Adjustment over the landmark points and build a *feature map* as a cloud of 3D points.

Figure 2.6 shows a MRF regarding the Keyframe method.

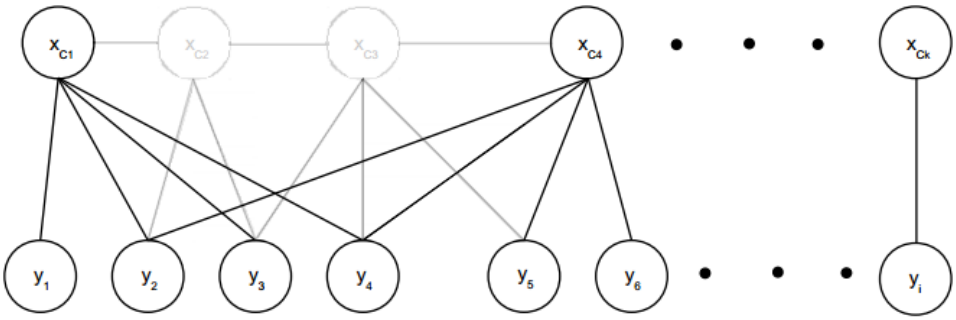


Figura 2.6: *SLAM MRF Keyframe based.*

According to this approach, the poses not identified as keyframes are deleted with their edges.

The *Filtering based SLAM* 2 consists essentially of a *pose marginalization* process introducing extra dependencies over the landmarks.

Figure 2.7 shows a MRF regarding the Filtering method.

The marginalization sums up the information about the pose sequence in just one random variable X_{C_k} *last position* but it requires the creation of many new edges:

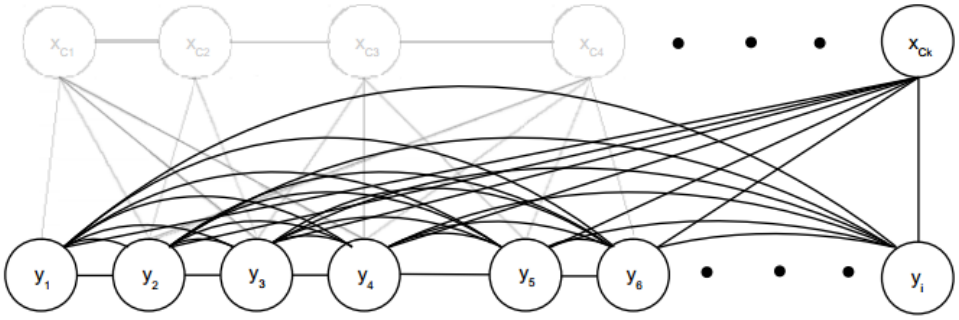


Figura 2.7: *SLAM MRF Filter based.*

not only the last position is connected to the whole sequence of observed landmark position but moreover these ones are connected among them.

Analyzing figure 2.6 and figure 2.7 it is possible to see that the keyframe method produce a more sparse structure hence it is cheaper from a computational point of view than the filtering method. On the other hand the filtering method produces a $P(\{X\},\{Y\})$ *Joint PDF* between poses and observation that is more efficient.

The solution of an online SLAM problem is performed by means of a *Recursive Bayesian Filter* so to compute the *Joint PDF*.

2.5 Markov Random Fields - Theory

Basic Definitions

A Markov Random Field is a specific Graphical Model, more specifically it is an **undirected graph** where each node is associated to a Random Variable and each edge represents statistical dependency. They are different from Bayesian Networks because the latter are exclusively DAG (Direct Acyclical Graph).

Let $G = (V, E)$ be an undirected graph then let's identify with $N = |V|$ the number of nodes in the graph and let $X = \{x_i\}_{i=1,\dots,N}$ be a discrete finite set of Random Variable associated to the graph nodes by means of an invertible relation.

Let I_V be the set of the node indexes, expressed as natural number.

The set E of the edges defines the topology of the graph and from a *local perspective* it induces the fundamental concept of *neighbourhood* that can be represented by the definition of a funcion $I_{NN}(i) = \{j\}$ which associates to each node identified by i a group of nodes whose identification number is in the $I_{NN}(i)$ discrete finite set of indexes that are then directly connected to the initial one hence

$$I_{NN}(i) = \{j | (i, j) \in E\} \qquad \forall i, j \in I_V \tag{2.25}$$

The topology of the graph defines the way the random variables are statistically related. Such a graph is called a *Markov Random Field* if the random variables satisfy one of the *Markov Properties*

Pairwise Markov Property Let $X_i, X_j \in \mathscr{X}_G$ it holds that if the 2 variables are not neighbours then they are conditionally independent.

Let's recall that conditional indepence means that

$$P(X_i \cap X_j | \{X_k\}_{k \in I_v \setminus i,j}) = P(X_i | \{X_k\}_{k \in I_v \setminus i,j}) P(X_j | \{X_k\}_{k \in I_v \setminus i,j}) \tag{2.26}$$

In general MRFs used in computer vision applications consist of a *double layer structure* as shown in Fig. 2.8 consisting of a

- *Data Layer* represented by a graph whose structure is a square lattice with no connections among its nodes, containing the Input Data namely the pixel values of the original image
- *Label Layer* represented by the above presented MRF graph containing the dynamic and the Output of the processing

Since now on, if not differently stated, the Y will identify the Data Layer state hence the Y_i will identify a particular Data Layer element state and X will identify the Label Layer state hence the X_i will identify a particular Label Layer element state.

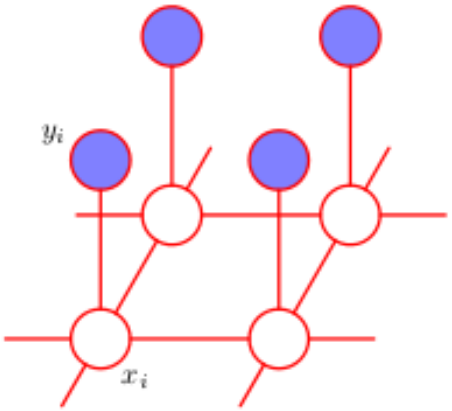


Figura 2.8: *MRF double layer structure. Blue filled nodes represent the Data Layer and White filled nodes represent the Label Layer with a classic square lattice structure endowed with 4 neighbours topology (Image from Bishop [2])*

Measure induced over the State Space The exact solution of a MRF regards finding $P(X|Y)$ *Measure* hence the Global Conditional PDF (Probability Density Function) induced by the Local statistical relations defined over the system.

The simplest strategy to compute an approximation of the unknown global PDF is to perform a Monte Carlo Sampling of the State Space but this strategy is not applicable in MRF context: because of the huge size of the state space the number of samples needed to get meaningful information is too big. Actually considering Ω the State Space, with $X = \{x_i\}_{i=1,\dots,N} \in \Omega$ a generic state of the MRF, with $Y = \{y_i\}_{i=1,\dots,N}$ the measurements hence the image and with $L = \{l_i\}_{i=1,\dots,m}$ the label set, it is immediate to observe that this value becomes very large even considering small systems: let's think about a $|L| = 2$ binary classifcation over a 10×10 MRF and hence image, then the cardinality of the space is

$$|\Omega| = 2^{100} \sim 10^{30}$$

To address this issue, proper algorithms have been developed like the Metropolis Hastings [14] and the Gibbs Sampling [15].

Regarding the global PDF a very important theoretical achievement has been brought by the Hammersley - Clifford Theorem [16] which states that the global PDF is of Gibbs type hence

$$P(\{X=\sigma\}|\{Y=\sigma_Y\}) = \frac{1}{Z(\sigma_Y)}\exp\left(-H(\sigma,\sigma_Y)\right) \tag{2.27}$$

where $\sigma \in \Omega$ identifies a specific point in the phase space, σ_Y identifies a certain measurement result and $H : \Omega \times \Omega_Y \to \mathbb{R}^+$ identifies the Hamiltonian of the System.

Finally $Z(\sigma_Y)$ is the *Partition Function* that depends only on the σ_Y input data because it is just defined as the integral of the Gibbs Measure over the State Space

$$Z(\sigma_Y) = \sum_{\sigma\in\Omega}\exp\left(-H(\sigma,\sigma_Y)\right) \tag{2.28}$$

From a numerical point of view it is important to observe that the straight computation of this term in the MRF context is almost always not possible because of the huge cardinality of the State Space Ω even considering very small systems.

Further the H.C. Theorem states that the Global PDF factorizes over the graph cliques (complete subgraphs) hence

$$P(\{X=\sigma\}|\{Y=\sigma_Y\}) = \frac{1}{Z(\sigma_Y)}\prod_{c\in\mathscr{C}}\exp\left(-H_c(\sigma_c,\sigma_{Y,c})\right) \tag{2.29}$$

with $\mathscr{C}$ Set of all of the MRF graph cliques, σ_c the subset of nodes involved in a certain clique and $\sigma_{Y,c}$ the subset of measurements that interact with the previously mentioned nodes.

This kind of result leads to an interesting energetic consideration: the Hamiltonian is essentially composed of n categories of terms where n is the highest order of the cliques in the graph hence the graph topology defines the Hamiltonian Function structure.

The energetic effect of the *external measurements* are also regolated by the topology: in typical computer vision applications it's common to correlate each image pixel with a MRF node in a one-to-one fashion hence the image induces a **potential field** on the MRF influecing the endogenous dynamic.

Hamiltonian The Hamiltonian Function is the core of the MRF method: this function contains all the information related to the behaviour (hence the evolution) of the system.

In general, the Hamiltonian represents in a quantitative way how the (partially) conflicting goals are combined.

Typically two main (partially conflicting) goals are pursued at the same time

- minimizing the likelihood (hence the similarity measure between the measure and estimated label)
- minimizing the *prior knowledge* vincula

The modeler choices then regard defining these two Hamiltonian subfunctions.

Usually the likelihood function is defined as a similarity measure in a feature space that's generally diffemorphic to an euclidean space, the euclidean distance is a common choice.

The prior knowledge vincula are usually represented by means of pairwise interaction terms between a pair of nodes (but it is obviously possible to define even more complicated relationships involving at least 3 nodes at a time) and if the relationship involves only neighbouring nodes, the graph respects a spatial Markov property and it is called Markov Randon Field.

Even if the likelihood function is simple, like a continuous convex similarity measure, the second term is generally not so easily tractable because each node is influenced by its own neighbourhood. This leads to complex energy function shapes and their minimization is in general a NP Hard problem requiring custom approximation strategies for real applications.

Topology and Hamiltonian Structure The Hamiltonian Function represents the essence of the MRF Modeling and its minimization leads to the labeling problem solution.

The structure of the Hamiltonian Function depends on the graph topology hence on the *contextual constraints* that need to be represented.

The most common topology categories are *grid like* and *parts based*.

Grid like topologies In **grid like** topologies a bijective relation among the Data Layer elements and the Label Layer elements is present. The structure of the Label Layer is then the same of the Data Layer namely a regular square lattice (as the Input image) but the neighbourhood has to be defined.

The Hamiltonian Function induced by a typical **grid like pairwise topology** is as follows

$$H(\sigma) = \sum_{i \in \mathscr{D}} \phi_i(x_i, y_i) + \sum_{l=\langle i,j \rangle} \varphi_l(x_i, x_j) \tag{2.30}$$

It means that this MRF model consists of

- $\{\phi_i\}$ $i = 1, ..., N$ singleton terms that represent the potential field derived from the Input Image into the MRF dynamic
- $\{\varphi_l\}$ $l \simeq 1, ..., 2N$ pairwise terms that represent the endogenous dynamic

Hence the first category of terms represents *data consistency constraints* and the second category of terms represents *contextual constraints* like smoothness.

The definition of the functions composing the Hamiltonian depends on the particular application.

Parts based topologies In **parts based** topologies the Label Layer Graph is defined according to a certain *prior model* searched in the image. This model sets both the structure and the neighbourhood of each node.

Typical applications regard performing *Object Recognition* in the image, like face recognition hence the parts model represent the structure of the face, in terms of features, searched in the image.

The Hamiltonian structure of a parts based topology depends strongly on the particular application.

Bayesian Framework It is possible to observe the MRF Framework from the Bayesian perspective considering the Bayes formula

$$P(X|Y) \sim P(Y|X)P(X)$$

From a mathematical point of view it consists of integrating the $P(Y|X)$ *Likelihood* all over the State Space using the $P(X)$ *Prior* as *probability measure* on this space.

By means of H. C. Theorem it is known that

$$P(X|Y) \sim \exp\left(H(\{x\};\{y\})\right)$$

hence the *Global Hamiltonian Function* can be decomposed as follows

$$H(\{x\};\{y\}) = H_{likelihood}(\{x\};\{y\}) + H_{prior}(\{x\})$$

Hence the Bayesian concepts of likelihood and prior correspond to parts of the global Hamiltonian Function. The energetic contribution given by the likelihood function is related to a *distance* between the measured noisy state and the internal state, while the energetic contribution given by the prior is related to contextual vincula.

Integrating the Bayesian perspective with MRF framework allows to express the Posterior Distribution in terms of Hamiltonian Function hence a new framework is defined, called *MAP MRF* namely *Maximum A Posteriori Markov Random Field* where X represents the internal state of the MRF, Y represents the measurements and the goal is to compute the $P(X|Y)$ Posterior PDF.

Energy Minimization By means of the Hammersley Clifford Theorem, solving the MAP MRF inference problem can be considered equivalent to the computation of the minimum energy state.

Unfortunately the Hamiltonian is in general a non convex function with a lot of local minima hence this minimization problem is not trivial and actually it has been demonstrated to be NP Hard [17],[18] and different algorithms have been proposed.

ICM The Iterated Conditional Models (ICM) algorithm is iterative: at the k iteration one single node is considered and the following problem is solved

$$x_i^{(k)} = \arg\max_{l \in L} P(X_i|X_j, Y_i) = \arg\min_{l \in L} H(x_i; x_j, y_i) \quad j \neq i$$

computing $x_i^{(k)}$ the label to assign to the i node to minimize considering only x_i as variable and all th rest as parameters.

It is a typical deterministic gradient descent method that converges very fast but it is not able to escape from any local minimum hence it is very sensitive to the initial conditions.

The ICM has been used by Besag et al. [19] as the first solution strategy.

Simulated Annealing Simulated Annealing is a probabilistic algorithm that mixes downhill movements with uphill movements depending on a certain the value of a *temperature* parameter making it able to escape local minima.

With a high value of temperature the system is in an ergodic state and the stochastic dynamic is the dominant one while slowly cooling the system (reducing the temperature parameter) reduces the uphill movements favouring the possibility to fall in the global optimum.

The performance of this algorithm are sensitive to the initial temperature (if it too low also to the initial state) and the cooling schedule with respect to the Hamiltonian function profile.

In fact, Simulated Annealing shows convergence to the global optimum only from a theoretical point of view with infinite time but in practical applications the results depend significantly also on the shape of the Hamiltonian function.

The Simulated Annealing has been used by Greig et al. [20] as an improvement with the respect to the previous strategy.

Belief Propagation Belief Propagation or equivalently Max Sum Product is an algorithm that solves inference problem defined over a probabilistic graph hence also on MRF [21].

This algorithm is very interesting because the theoretical study of its solutions led to demonstrate the equivalence among them and the minimum *Free Energies* hence allowing for the exploitation of the theoretical results, developed in the statistical physics field, in terms of closed form approximate solutions.

The Free Energy is a well known concept initially developed in the statistical physics environment and it can be expressed synthetically as follows

$$G = H - TS$$

where G is the Gibbs Free Energy, H is the Enthalpy, T is the Temperature and S is the Entropy.

In the context of Information Theory a similar result can be obtained computing the Kullback-Leibler Divergence between the real distribution solving the problem (in case of MRF the H.C. Theorem it is a Gibbs Distribution which is defined on the basis of the Hamiltonian Function of the System) and an approximated (typically more tractable) one.

Minimizing the Free Energy hence means minimizing the *distance* (actually the K.L. divergence is not a real distance but anyway it shows interesting properties) between the real distribution and the approximated one.

This kind of framework leads to the definition of *Variational Inference* problems where a certain distribution (showing interesting properties to solve the original graph related inference problem) is chosen to approximate the real distribution and its parameters are adjusted while minimizing the free energy hence considering $P(X;\sigma_p)$ as the real distribution with σ_p its known parameters vector and $Q(X;\sigma_q)$ as the approximated distribution with σ_q its unknown parameters vector the problem is as follows

$$\tilde{\sigma}_q = \arg\min_{\sigma_q \in \Omega q} D_{KL}(P,Q) = \arg\min_{\sigma_q \in \Omega q} \mathscr{G}(P(x;\sigma_p), Q(x;\sigma_q))$$

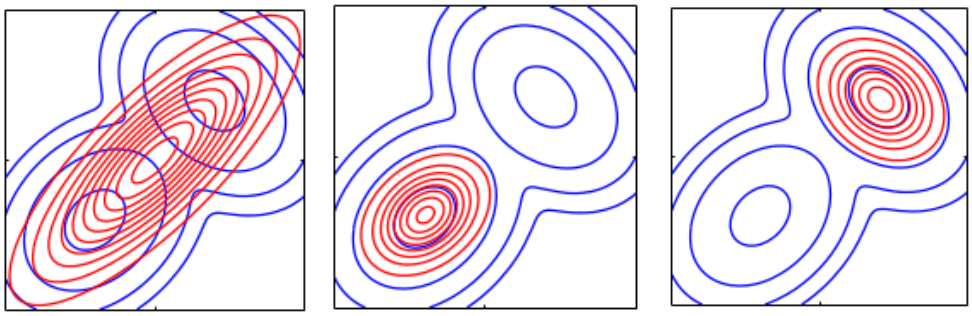
with $\mathscr{G}$ functional computing the free energy.

The quality of the approximation hence significantly depends on the choice of the approximating function.

For example if the original PDF is bimodal and the approximating function is uni-modal there is no parameters setup that can lead to a global approximation: the Variational Inference problem will hence show two minima, each one connected to a mode of the original distribution.

Let's recall that the Kullback-Leibler divergence is not a real distance because it is not symmetric hence using $D_{KL}(P,Q)$ leads to a straight KL divergence problem

while using $D_{KL}(Q,P)$ leads to a reverse KL divergence problem and they provide conceptually different solution as shown in Fig. 2.9.



(a) (b) (c)

Figura 2.9: Direct KL(a), Reverse KL for one mode (b), Reverse KL for the other mode (c) (Image from Bishop [2])

Straight KL aims at finding global optimum for the approximating function but in case the shapes are inconsistent the result is not good. Reverse KL aims at finding local optima with the approximating function fitting the original distribution very well only in certain points.

A noticeable approximation that allows for a closed form solution of the inference problem is the **Mean Field** approximation.

It is characterized by choosing an approximating distribution that considers every node as **statistically independent** from all the others because the pairwise interactions are replaced by singleton interactions with a *mean field term* so the PDF becomes

$$P(X_i|\{X_j\}_{j\in I_{NN}(i)},Y) = \tilde{P}_i(X_i|Y_i) \tag{2.31}$$

hence leading to a Joint PDF factorizing over each single node as follows

$$\tilde{P}(X|Y) = \prod_{i\in\mathscr{D}} \tilde{P}_i(X_i|Y_i) \tag{2.32}$$

A factorizing PDF is easily minimizable because the independence property splits the global minimization in a set of $|\mathscr{D}|$ independent problems easily tractable.

Certainly this is a strong assumption and provides good results only in a specific subset of real problems.

Graph Cut The general framework regarding the use of Graph Cut Algorithm to solve MAP inference problems on probabilistic graphs has been proposed by Greig [20].

The minimization problem on the graph can be trated as a max flow/minimum cut one as shown by Boykov et al. [22].

Considering the $G = (V,E)$ Graph containing the dynamic of the MRF, a $C = (S,T)$ Graph Cut is a partition of the graph space, realized removing a certain set of edges, that divides the nodes into two distinct groups such as $S \cap T = \emptyset, S \cup T = G$

Considering $C = \{e_i = (u,v) | u \in S, v \in T\}$ the set of the edges involved in a certain cut, it is possible to define the *weight of a cut* as the number of the edges in C for an unweighted graph and as the sum of edges weights in C for a weighted graph.

The Image Segmentation problem gets treated by the Graph Cut framework as depictied in Fig. 2.10.

All the above mentioned problem transformations preserve its complexity hence the minimum cut is still NP Hard so the goal is to compute good approximations in less than exponential time.

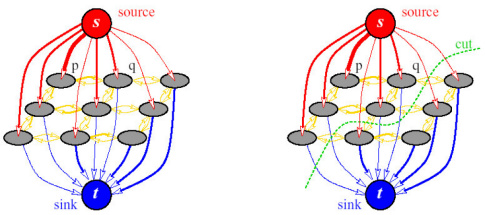


Figura 2.10: *Image Segmentation performed on MRF by means of Graph Cuts*

Graph Cut methods proved very interesting in practical applications because of the high quality of their solutions computed in short time, the drawback lies in the fact that unlike Simulated Annealing, which can be used to minimize a general energy function, the Graph Cuts method works only on certain energy function as described in Kolmogorov [23].

2.6 Markov Random Fields - Computer Vision Applications

In the following section, some noticeable applications regarding the above presented theory are provided.

Image Restoration

Image Restoration is one of the most common applications for MRF.

Image prior is represented in the Hamiltonian terms not related to the likelihood function.

Essentially using the MRF framework requires solving 2 major categories of consistent problems

- defining the graph topology and the potential functions
- setting the parameters

In early approaches it is common to observe simple topologies and potential functions as well as a modeler defined parameters setup.

In recent approaches it is possibile to observe more complicated topologies and potential functions (Higher Order MRF) and methods for automated parameters learning.

Applications Among the first important applications of MRF to the image restoration problem it is possible to mention Besag et al. [19] where a MRF has been used to model the contextual vinculus of the *neighbourhood similarity*.

However this methods suffers a common problem among these early applications: the corresponding Hamiltonian had no edge preserving terms hence leading to an *oversmoothed solution*.

Another important limitation concerning the proposed methods regards the minimization strategy: when an ICM algorithm is used the computed solutions are very likely to be local minima.

Hereinafter Figuereido et al. [3] achieved the result shown in Fig. 2.11 encoding in the Hamiltonian Prior subfunction the dual goal of smoothing (fourth image from the top) and discontinuity preserving (third image from the top).

Certainly decideding how to combine these dualistic goals is not trivial and strongly influences the quality of the result.

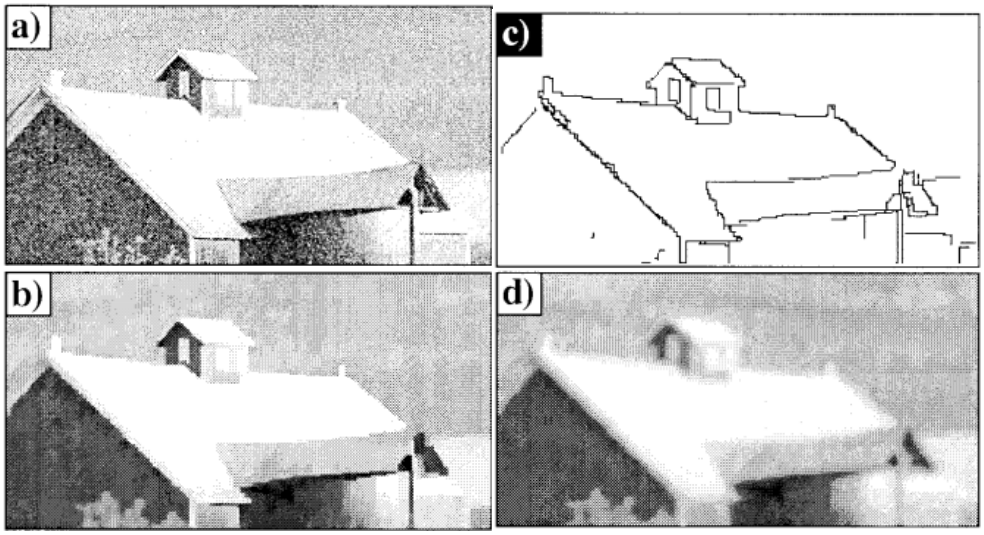


Figura 2.11: *Figuereido et al. [3]. (a) is the noisy version, (b) is the restored version considering the discontinuity preserving, (c) just shows the discontinuities hence achieving a sort of segmentation and (d) shows the purely smoothed image.*

More recently Babacan et al. [4] got the result shown in Figure 2.12 using a struc-

tured approach exploiting the Baysian Framework, the GGMRF (General Gaussian MRF) and the Variational Framework.

The GGMRF is used as image prior and it depends on α, β hyper parameters considered as random variables (hence they also need to be estimated) following a certain distribution. The Gamma distribution, depending on the params $a_\omega^0, b_\omega^0, \omega = \{\alpha, \beta\}$ is then used as hyper distribution for these hyper params but unfortunately no specific justification is provided for this choice.

The full posterior PDF hence contains 3 random variables namely X the unknown image and α, β hyper parameters and it is defined with respect to the global Joint PDF containing these variables and Y the observed image.

Considering this posterior PDF is not tractable in closed form, a certain family of solution is investigated: solutions that factorize the hyper params block and the unknown image block, and parameters estimation is performed using the Variational Framework.

More recently Sun et al. [24] proposed an interesting approach based on a more structrured version of MRF namely a non local-rage MRF (NLR-MRF) and a gradient-based learning method to learn the potential functions.

Markov Random Fields for Image Segmentation and Labeling

The problem of image segmentation and labeling is a fundamental one in Computer Vision. The goal is to assign to each pixel in the image a certain label from a known discrete finite set $L = \{l_1, l_2, ..., l_k\}$.

Early approaches to Image Segmentation make wide use of thresholds as reviewed by Sahoo et al. [25] and Lee et al. [26] but certainly this is a too specific kind of solutions that relies too much on the modeler choices.

PDE methods improved significantly this field, providing a less specific approach to the problem. Some noteworthy contributions are the active contour/snake models [27], [28], [29] and anisotropic diffusion based ones [30].

MRFs have been introduced in Image Processing by Geman et al [15] and proved effective while computationally onerous.

(a) Initial Image

(b) Adding Gaussian Noise

(c) Restoration result

(d) Another restoration result

Figura 2.12: Babacan et al. [4].

One of the first attempts to perform data clustering, hence segmentation, on a graph by means of combinatorial graph cut is due to Wu et al. [31].

As a consequence of the available computational power increase, this kind of methods gathered attention and started to be considered promising [32].

Among their first applications, they have been used for texture models as by Cross et al. [33] and Derin et al. [34].

An interesting contribution has been provided by Lakshmanan et al. [35] that investigated the problem of performing simultaneous parameters estimation and segmentation, solving the MRF with Simulated Annealing, while Manjunath et al. [36] compared thier approach with the previous one, concluding that separating estimation from segmentation significantly simplifies the problem and in general leads to better and more computationally manageable results.

Applications One of the most important contribution regarding the application of MRFs to Image Segmentation has been brought by Shi et al. [37].

The authors proposed a radical perspective shifting in approaching this kind of problems: they started observing the essential ill posedness of the segmentation problem (actually it does not exist a unique correct answer but more than one, depending on the chosen focus) hence they pointed out to approach it in the Bayesian Framework so to be able to model the *chosen point of view* by means of the *prior function*.

Furthermore the authors observed that the expected solution should not be a "flat" segmentation but a hierarchical one hence a tree or more precisely a *dendogram*.

After this very important introduction, the authors started presenting the tools to achieve the goal: MRF and variational formulation.

Another important work is the one provided by Boykov et al. [5] consisting of the application of a combinatorial optimization technique on graphs namely the *s-t graph cut* (a technique succesfully used to deal with other important computer vision problems like Image Restoration, Multi View Reconstruction, . . .) to problem of background / foreground segmentation for photos (see Figure 2.13), videos editing (see Figure 2.14).

Figura 2.13: *Application of the Boykov et al. [5] algorithm to photo segmentation*

MRFs have been used in the field of Image Segmentation because they tend to perform better than other methods like the active contour/snake one or deformable shape matching one, presented in Felzenszwalb et al. [38], especially in presence of noise and occlusions.

Figura 2.14: *Application of the Boykov et al. [5] algorithm to video sequence segmentation*

An interesting contribution has been brought by Vu et al. [6] using MRF with pairwise topology, introducing priors regarding different shapes and solving by means of Graph Cuts following Energy Minimization strategies proposed by Boykov et al. [39].

Examples of priors are shown in Figure 2.15 and results are shown in Figure 2.16

Figura 2.15: *MRF priors introduced in Vu et al. [6]*

MRF can be succesfully integrated with other methods to improve them.

For example Chen et al. [40] developed an interesting Image Categorization approach and observed that the integration of a MRF to model contextual information would have improved the system.

An interesting recent approach is the one presented by a KaradaÄ§ et. al [41]: the authors propose a MRF based framework to fuse the outcomes deriving from bottom up unsupervised segmentation with top down segmentation depending on a (possibly human) *domain expert* prior, providing *domain specific information*.

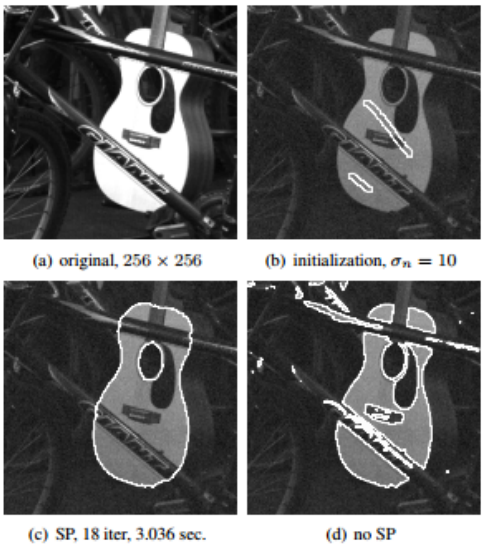


Figura 2.16: *Image Segmentation results performed at different level of the parameters in Vu et al. [6]*

MRF in Stereo Matching and Multicamera Scene Reconstruction

Stereo Matching has been a very active field in Computer Vision for a long time.

Common methods to solve this kind of problems formulated by means of the Bayesian Inference over MRF in (almost theoretical) papers in early vision essentially proposed Simulated Annealing as solution method and this led to poor performance in real applications.

An important contribution has been brought Roy and Cox [42] that introduced the Graph Cuts as a new solution strategy. They have been able to carry out sucessful real applications and they observed better performance with respect to the previous working method (line-by-line stereo matching) in reasonable time despite no theoretical guarantee.

Roy [43] solved such a problem by means of Max Flow/Min Cut but not using a

discontinuity preserving energy function he got an oversmoothed solution.

Scharstein et al. [44] proposed a valuable review regarding methods that compute depth maps with respect to a certain image plane.

This kind of problems can be formulated in Bayesian MRF framework and solved efficiently by graph cuts or loopy belief propagation algorithms.

Applications Kolmogorov et al. [7] developed a graph cut based approach for multi camera scene reconstruction, taking into account a variety of important aspects previously mentioned like discontinuity preserving energy function, photoconsistency and visibility. Results are shown in Figure 2.17.

Figura 2.17: *Depth Image by Kolmogorov et al. [7]*

Vogiatzis et al. [8] developed an idea of Boykov and Kolmogorov [45] so to perform multiview volumetric scene reconstruction by means of the MRF-Graph Cut Framework.

The original work was related to the Image Segmentation domain and the application to the multiview volumetric scene reconstruction one has not been immediate: dealing with occlusions and the formulation in that framework represented major issues, to solve which the concept of *base surface* [46] has been developed. Results are shown in Figure 2.18.

Figura 2.18: *Volumetric Reconstruction by Vogiatzis et al. [8]*

2.7 Bundle Adjustment

The bundle adjustment problem in computer vision can be summerized as follows: considering the same world point, of unknown coordinates, is observed by N different positions, so as many images are available, estimate the point world coordinates.

Let's introduce the following formalism

- $X \in \mathbb{R}^4 \quad X = (x, y, z, 1)^T$ the homogeneous world coordinates of the unknown point
- $\{x_i\}_{i=1,\dots,N} \quad x_i \in \mathbb{R}^3 \quad x_i = (u_i, v_i, 1)^T$ the homogeneous image coordinates for the point observed by the 2 cameras
- $\{P_i\}_{i=1,\dots,N} \quad P_i \in \mathbb{R}^{3\times 4} \quad P = (p_1, p_2, p_3)^T$ the camera matrixes

The equation that represents the relation between the world point and the image point is the following one

$$\lambda_i x_i = P_i X \qquad \forall i = 1, \dots, N \tag{2.33}$$

In real computer vision noise is present hence this relation does not perfectly hold. Then Maximum Likelihood are searched under the assumption of some Noise Model.

In order to setup an estimation problem a *cost function* needs to be defined, depending on some sort of error, so let's define the *reprojection residual vector* related to one image, depending on the X unknown 3D coordinates of the point

$$r_i(X) = \left(u_i - \frac{p_1^T X}{p_3^T X}, v_i - \frac{p_2^T X}{p_3^T X} \right)^T \tag{2.34}$$

Each component of the *residual vector* 2.34 represents the difference the *predicted image position* for the world point assuming its world position and the *observed image position*.

The cost function will associate to each *residual* a *cost*. Under the assumption of Gaussian Noise, the best cost function to perform a MLE (Maximum Likelihood Estimation) is the *sum of squared residuals* defined as follows

$$f(X) = \sum_{i=1}^{N} ||r_i(X)||_2^2 \tag{2.35}$$

Then the 3D coordinates estimation is obtained solving the following minimization problem

$$\hat{X} = \arg\min_X f(X) \tag{2.36}$$

Hartley and Zisserman [47] demonstrated that the minimization of the function 2.35 is a non convex optimization problem hence the use of a Gauss-Newton method could lead to be trapped local minima.

Different algorthms exist to deal with this problem and essentially they performs two kind of tasks

1. Identifying the *descent direction*
2. Computing an estimate for the best *step size*

The *descent direction identification* 1 is usually performed exploiting the Jacobian of the residual function.

Let's consider a local to x_0 approximation of the residual function

$$r(X_0 + \delta X) \simeq r(X_0) + J(X_0)\delta X \tag{2.37}$$

So a local to x_0 approximation of the cost function becomes

$$f(X_0 + \delta X) \simeq ||r(X_0) + J(X_0)\delta X||_2^2 \tag{2.38}$$

So the $\delta\hat{X}$ *best descent direction* is the solution of the following optimization problem

$$\delta\hat{X} = \arg\min_{\delta X} f(X_0 + \delta X) \tag{2.39}$$

considering that

- X_0 is given at the beginning of the problem (previous position in the iterative method)
- δX is the objective of this minimization

The *step size computation* 2 can be performed in different ways and essentially three types of algorithms exist:

- Linear Search
- Levenberg-Marquardt
- Trust Region

In the context of Bundle Adjustment the method that is commonly used to solve this the problem is the *Levenberg-Marquardt* algorithm. It consists of an interpolation between the Gauss-Newton algorithm and the gradient descent algorithm. This algorithm has shown the capability to converge quickly from a wide range of initial guesses. However it does not deal with the local minima problem hence the *initial estimate* is important for the quality of the final estimation.

The *initial estimate* is computed solving the linearized approximation of the original non linear problem.

The linear solution conveys the distorsion introduced by the linearization procedure however using at as initial point for the LQ algorithm usually produces good enough results.

Finally the *error propagation* techniques can be used to compute a *confidence measure* for the estimated solution.

The last in observation regard the fact that least squares methods are non robust hence sensitive to the outliers presence.

2.8 Location recognition

In computer vision, the location recognition problem consists in deciding if a certain observed place is already present in a certain database.

More formally, the following elements are defined

- $\{s_i\}$ a set of images (at least one) representing a certain place
- M a map containing information about a set of places

As it has already been introduced in section 2.3 the concept of map needs to be detailed: different kind of maps exist, representing the environment at a different abstraction level.

If the Database consists of *metric map* hence a set of images associated to places, it becomes an *image searching* problem.

Early approaches to *image searching* consisted of performing *linear features matching* between the features set extracted from *s query image* representing a place and the features set extracted from the images in M dataset.

Considering the first implementations of this strategy have been realized on standard commercial PC it has been observed working quite well for small datasets, consisting of some thousands of images, but showed scalability problems when switching to larger datasets of millions of images.

Approaches like [48], [49] and [50] used Database consisting of less than 1000 images but the order of magnitude that is necessary to consider for SLAM applications is at least of millions of images.

Let's consider that from a normal place image it is possible to extract from hundreds up to about one thousand features and considering a one million images datasets, about one billion descriptors should be checked for each feature extracted from the query image: this will lead to poor performance (about some seconds for each query) on the above mentioned hardware.

Carrying out efficient searches (order of magnitued of tens of milliseconds for example) in large datasets on commercial hardware requires the development of new strategies: the problem of *location recognition* becomes actually a *scalable location recognition* problem.

In general, approaches aiming at improving scalability essentially rely on the introduction of *higher levels of abstraction* in the dataset representation as well as indexing techniques.

This abstraction process consists in summing up the information so to create smaller search space that however preserves the most relevant information. Moreover the abstracted representation shows very useful inavariance properties.

Interesting strategies that have been developed make use of *vocabulary search trees* and they are described in section 2.9.

2.9 Visual Vocabulary Tree

The *Visual Vocabulary Tree* is a ***Local Feature based retrieval method***.

The concept of *visual vocabulary tree* has been inspired by the strategies developed in the *text search and retrieval* community.

It concerns the problem of effectively and efficiently searching across large datasets. This kind of problem has first been studied in the *text search* community because of the early availability of such datasets, the *computer vision* community then followed some of these ideas.

In the *textual document* context, one problem with important implications for searching, regards the ability to *summerize a document* hence associating it with a *semantic* information.

A common approach is the so called *bag-of-words* which consists of counting keywords identified in the document and then building a sort of *document semantic fingerprint* by means of the keywords frequency distribution.

In the computer vision context, images are analogous to documents hence image features could be considered the analogous for words. Essentially this is not precisely true because features are equivalent to single points in the *feature space* which is usually a high dimensional possibly real valued space.

In order to build a true *visual word* analogy, an abstraction has to be imposed on such feature space. Such an abstraction can be obtained by means of *space quantization*.

The innovative approach presented by Nister et al. [51] proposed the idea to *learn* the *vocabulary of visual words* implicitly associated to a certain dataset by means of unsupervised methods.

Assuming to deal with $D = \{s_i\}_{i=1,\dots,N}$ a large image dataset, for each s_i image in the dataset a set of features (e.g. SIFT) is extracted

$$\{f_j^{(i)}\}_{j=1,\dots,N_f(i)} \qquad f_j^{(i)} \in \mathscr{F} \forall i, j$$

with

- $N_f(i)$ Number of features extracted from the i image
- $\{f_j^{(i)}\}$ set of features related to the i image
- $\mathscr{F}$ high dimensional feature space

hence that image is represented by a set of feature points in the corresponging high dimensional feature space and the whole dataset is represented by the union of the single clouds of feature points.

The goal is to impose an abstraction on the feature space depending on the D training set and it happens performing a *hierachical feature space quantization* by means of an iterative *k means clustering*.

The algorithm paramaters are the following ones

- k k means clustering main parameter

- L number of iterations

At the beginning of the algorithm, the first k means clustering is performed on $(\mathscr{F}, D)$ and k centroids coordinates are computed then the space is partitioned according to a Voronoi Diagram based on the previously defined k centroids.

At the following iteration, each of the k Voronoi regions is again partitioned as previously explained.

This process builds a *tree structure* for space partitioning and each leaf can be considered a *visual word.*

The tree structure can be stored in data structure like

$$\left\{\left\{c_{i,j}\right\}_{j=1,\ldots,k}\right\}_{i=1,\ldots,L}$$

with

- $c_{i,j} \in \mathscr{F}$ centroid coordinates in the feature space
- i counting over the L depth levels
- j counting over the k regions for each depth level

The k parameter can be considered the *branching factor* while the L parameter can be considered the *depth factor* of the tree structure hence the vocabulary consists of k^L visual words.

A generic $f \in \mathscr{F}$ feature point can then be easily associated with a visual word: the feature point information are used to navigate the vocabulary tree until the final leaf hence the corresponding visual word.

Essentially this is an iterative process described by the following algorithm

Algorithm 1 Visual word association to a feature point

for $i = 1$ to L **do**
  $\hat{j} = \arg\min_{j=1,\ldots,k} \left\| f - c_{i,j} \right\|$
  Move inside the $\hat{j}$ region
end for

at the end of the algorithm the $\hat{c}_{i,j}$ centroid resulting from the last operation is the globally closest one to the f feature hence an association with a *visual word* has been defined.

Let's observe that in real applications the above defined algorithm gets slightly modified as follows

Algorithm 2 Visual word association to a feature point - Real Algo

for $i = 1$ to L **do**
  $\hat{j} = \arg\max_{j=1,\ldots,k} f \cdot c_{i,j}$
  Move inside the $\hat{j}$ region
end for

The difference lies in the substitution of the distance function, identified as the norm of the difference vector, with the dot product. This is a widely used trick based on the observation that the distance between 2 points in a real vector space is maximum when their dot product is minumum and viceversa. Considering that dot products are less computationally expensive than differencing and norming, this strategy brings performance improvements.

This algorithm can be applied for all the features extracted from a generic image and a *visual bag of word* for that image can be defined.

Then a *similarity measure* is defined to compare an image in the database and a query image. This measure is naturaolly based on the *common set of words* shared between the images.

A specific *visual word* related to an image feature is uniquely identified by a *feature path* which is the $\{n_i\}_{i=1,\ldots,L}$ *sequence of nodes* in the vocabulary tree the feature meets before reaching a leaf.

The image similarity can then quantitatively defined in terms of *similarity of feature paths* across the vocabulary tree.

Furthermore, in a Bayesian fashion it is possible weight the nodes in different ways according on their discriminatory capability (depending on the feature set).

Finally an *indexing strategy* is defined to quickly identify a *subset of relevant images*.

For each $s_i \in D$ image in the Dataset, it is possible to define the *visual words frequency* as follows

$$v_{i,j} = \frac{N(i,j)}{N_{tot}(i)} \qquad j = 1,...,k^L \quad i = 1,...,|D|$$

with

- $v_{i,j}$ the frequency of the j visual word in the i image in the dataset
- $N(i,j)$ number of times the j visual word appears in the i image
- $N_{tot}(i)$ total number of visual words in the i image

This quantity allows for a definition of *visual word frequency ranking* for a certain image.

Computing for the query image the *visual word frequency ranking* it is possible to search the dataset index for the images with a similar ranking hence defining a *search subset* to be analyzed thoroughly by means of one to one comparisons between each image in the subset and the query image, using the above mentioned similarity distance.

To sum up, the *vocabulary tree* approach is a special case of the *bag-of-words* approach hence it shares the same limitations

1. Quantization involves loss of information
2. Spatial Informations about the features are not considered

Regarding the *information loss by quantization* 1 using better *scoring strategis* could limit this problem.

Reagrding the *spatial information loss* 2 the use *spatial verification* methods applied to two images with a high *bag-of-words similarity* is necessary to estimate e *spatial transformation* aimed at evalutaing the *features spatial consistency*. Typical methods are *RANSAC* and *Generalized Hough Transform*.

2.10 KD Trees

The *KD Tree* is an indexing technique that is used to realize a specific *Approximate Neareset Neighbor* (ANN) implementation for efficient *image retrieval* on large datasets.

A KD Tree is built iteratively partitioning the feature space each time into *two subspaces* according to a certain *splitting heuristic* which addresses the following issues at each iteration

1. dimension to split long
2. at which value split

Considering a k dimensional feature space, at a given iteration a certain *subspace* needs to split and a choice for the *dimension to split along* 1 needs to be taken. A common strategy is to compute the variance for the points population in the above mentioned subspace and split it in its components along the dimensions and then choosing the highest one.

The second issue is choosing the *value for the split* 1. A common strategy is to project the points population on the above defined dimension and computing the median so to have half of the population on one side and half on the other side.

To sum up the above described splitting algorithm for the KD Tree considering that

- N Number of total iterations needed to build the tree
- $s(i)$ Function that computes the unique tree traversing path for a certain node, considering the iteration it has been created

Each iteration starts with the $S_{s(i)}$ subspace related to a certain node tree and the strategy to compute the $\{S_{s(i),j}\}_{j=1,2}$ two subnodes is described. Then the algorithm forks and restarts at each subnode.

The splitting algorithm produces a $\{S\}_i$ *feature space partitioning* with i representing the unique position in the binary tree. The result is visually outlined in figu-

Algorithm 3 KD Tree Splitting Algorithm

Load $\{f\} \in \mathscr{F}$ Population of feature points
for $i = 1$ to N **do** ▷ Cycle over all the splitting iterations
Consider $\{f\}_{s(i)} \in S_{s(i)}$ ▷ Point Population in the considered subspace
Compute $\{\sigma_k\} = f(\{f\}_{s(i)})$ the k components of the variance along the dimensions
$\hat{k} = \arg\max_k\{\sigma_k\}$ ▷ Get the dimension related to the highest variance component
$\{f\}_{s(i)}^{(k)} = P_k(\{f\}_{s(i)})$ ▷ Project the point population on k dimension
$v = Median(\{f\}_{s(i)})^{(k)}$ ▷ Compute the median
Compute $S_{s(i),1}, S_{s(i),2}$ splitting along $\hat{k}$ at v value
end for

re 2.19, on the left the partitioned space is shown and on the right the corresponfing binary tree is reprsented.

The blue box on the left identifies half of the feature space hence it corresponds to half of the binary tree.

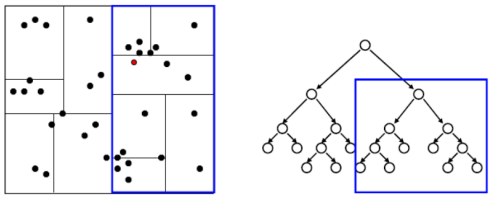


Figura 2.19: *A KD Tree example.*

This kind of structure improves the (approximate) nearest neighbor search performance reducing the number of comparisons needed.

Without this structure a linear search over tha whole dataset would be needed to solve the problem.

This structure essentially consists in a set of hyperplanes partinioning the feature space. Then a certain f *query feature* needs to be checked against a set of $\{p\}_{i=1,\ldots,L}$ *hyperplane vectors* by means of *dot product*. The sign of $f \cdot p_i$ identifies the relative

position of the point with respect to the hyperplane: positive on one side, nagetive on the other side, zero on the plane. Sequentially performing this computationally cheap operations the tree is traversed and finally the a leaf is identified.

A *tree leaf* is essentially a $S \subset \mathscr{F}$ *Feature Subspace* smaller than the initial one. By means of an indexing mechanism, the $\{f_i\} \in S$ *set of dataset features in the subspace* is easily identified.

Finally a linear comparison can be peformed between f *query feature* and $\{f_i\}$ computing the approximate nearest neighbor inside the leaf.

This method could fail in determining the real nearest neighbor because of the distrting effect introduced by the space quantization close to the borders.

For example let's consider the query feature is falling close the border of the leaf node. A linear comparison is performed among all the dataset features belonging to the same leaf node and a neareset neighbor candidate is computed but maybe, just over the subspace border in a neighbouring cell another dataset feature node exists with a smaller similarity distance. It is not discovered because the linear search is performed just in the identified cells and not in neighbouring ones.

Capitolo 3

Industrial Context

3.1 Brief Overview

Performing self localization and autonomous driving in the industrial envrionment is complicated by the absence of a widely used a widely used elsewhere technology, the GPS: the walls mask the GPS signal making it impossible for the receivers to detect it.

This condition has raised the necessity to investigate different technologies suitable for for indoor self localization.

An interesting work is the one by Torres et al. [9] the provides a list of different possible exploitable technologies summerized in figure 3.1.

In the context of the presente work, it has been decided to focus on the application of the RFID technology.

Essentially the idea regarded the development of a "smart floor" providing localization information the AGV can use to perform Data Fusion with other information sources like odometry.

The "smart floor" consists of a typical warehouse resin floor, with RFID Tags embedded. When each of this tag is interrogated, it answers back its Unique ID.

Once that floor has been realized, a calibration phase needs to be performed to associate each Unique Tag ID to a specific position, for example in terms of (x, y)

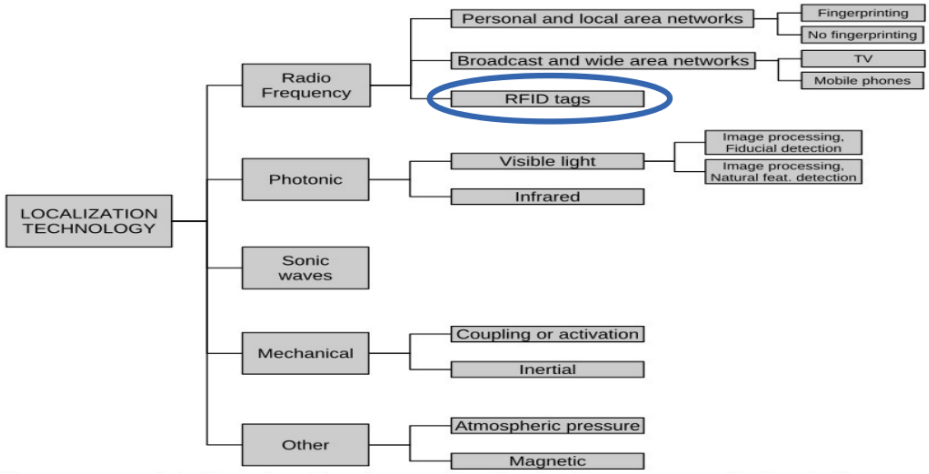


Figura 3.1: List of indoor localization methods provided by Torres et al. [9].

coordinates, in the warehouse.

Considering the floor (almost) never changes, the smart floor calibration needs to be performed only once.

To build the association between Tag ID and coordinates the following machine can be used: a moving robot equipped both with an RFID reader and laser scanners.

Assuming the RFID reader is able to read only one Tag at a time and that it reads only the closest one, when a Tag is revealed its position is estimated by the laser scanners readings.

3.2 Industrial Project

The research project has been developed in the context of an agreement with a company, which provided some of the needed infrastructures like the fork lift and the test space.

It consisted of a two phase project.

The first phase consisted essentially of a feasibility study performed by means of simulations.

The second phase consisted of a working prototype development, implementing a set of commonly agreed features.

3.3 Simulator

By means of simulations it is possible to analyze the system performance both from a qualitative and a quantitative point of view, for different parameters setup.

One of the main goals of the simulations regarded to assess the particle filter self localization sensitivity to a certain parameters set, containing the reading distance and tags density on the floor.

The latter parameters have a major impact on the solution full cost, which is indeed strongly influenced by the RFID number and typology.

The objective is then to be able to estimate the optimal RFID typology and density according to desired performance with respect to a certain solution budget.

Different simulations have been performed and some interesting results have been observed but unfortunately this analysis has not been thoroughly completed because the "smart floor" development has been put in stand-by because of company related issues.

3.3.1 Problem Definition

The general objective of this development phase is to build an analysis and testing framework relying on *simulations* for the self localization problem in the industrial environment with a specific focus on an RFID based technology.

The purpose of building a simulator mainly regards the possibility to study the proposed solution sensitivity to certain *solution parameters* exploiting a *model of the reality* thus without the need to frequently carry out expensive and time consuming real tests.

The parameters effect on the solution performance are interesting also because they condition the solution cost, both in economics and required time terms, as well.

3.3.2 Proposed Solution

The development roadmap has been defined as follows

1. Specific simulators development
2. Simulator testing
3. Simulators results are presented
4. Simulator integration

The step 1 depends on the fact that no need to develop a unique simulation framework has been identified in an early development stage hence different simulators focusing on specific situations have been realized.

The goal of the present phase consisted in the development of 3 software modules

1. An *Environment Simulator*
2. A *Data Analyzer*
3. A *Detailed Motion Analyzer*

The *Environment Simulator* 1 goal is to

- simulate the real environment with AGV moving in it, in order to perform a self localization aimed data collection, with the possibility to modify some key parameters
- visualize the online motion

A GUI will be realized to easily set up all the parameters and to visualize the motion while happening but all the generated data is logged to be further thoroughly analyzed by the *Data Analyzer* 2 module.

Note: for the first milestone, it has been decided to use in the *Environment Simulator* a holonomic vehicle model, instead of the more appropriate tricycle vehicle model. The reason for this choice regarded the desire to not introduce excessive complexity in an early development step so to keep the focus of this simulator on its goal:

the self localization performance assessment depending on the chosen technology and strategy.

The *Data Analyzer* 2 goal is to

- to load and process the log files generated by the *AGV Simulator*
- visualize synthetic data
- serve as platform to possibly carry out other detailed analysis

The *Detailed Motion Analyzer* 3 goal is to

- to load command sequence data to be sent to the steering wheel
- apply this information to a more precise model of the vehicle (with respect to the one used in the self localization aimed simulator)
- visualize the motion in a more precise way with particoular attention to the occupied space

This module has been added to perform a separate more precise analysis, if and where needed, of the vehicle motion with respect to the oversimplified one carried out in the *AGV Simulator* module.

In the step 2 some tests for the simulators have been carried out to verify the quality of their result and perform some parameters tuning when needed.

In the step 3 simulator results are presented regarding the following issues

- the self localization performance achieved by means of data fusion, performed by a certain algorithm, regarding different data sources and according to different parameters values
- the obstruction analysis while driving

The step 4 consisted of the development of an integrated simulator but this phase has not been carried in the context of the present thesis.

3.3.3 Development of the Simulator

Environment Simulator

The *Environment Simulator* has been developed in PYTHON because of the versatility of the language.

The simulator represents a "maze-like 2D world" where an AGV moves in.

This "flat world" (which essentially represented the smart floor warehouse) is essentially composed of 2 kind of blocks

- Free Space block
- Occupied block

The *free space* block is accessible to the AGV motion while *Occupied* block is not, furthermore RFID tags can only be present in the former block.

The AGV is required to move in this environment reaching a certain sequence of goal points, whose world coordinates are provided, hence for th AGV it is mandatory to be able to properly self localize in the world so to drive to next goal point.

While moving, the simulated AGV accesses the following source of information

- noisy wheel odometry
- noisy RFID Tag readings

The AGV motion model is the *2D holonomic vehicle* as previously explained.

The *self localization* is performed fusing the information regarding the vehicle noisy odometry and noise RFID observation using the **particle filter**.

It is possible to choose different *noise models* but regarding the Data showed in this thesis only the Gaussian Distribution has been used.

Particle Filter The following particle filter adaptation has been developed.

In the context of the *self localization problem* the Ω *state space* is the *set of all the available location* on the map (all the locations related to free space blocks).

At the beginning, the $P_0(x), x \in \Omega$ *prior position distribution* is sampled by N_p particles so the $\{\tilde{x}_i\}_{N_p}^{(0)}$ set of particles is generated. If no prior information is available, then the uniform distribution is used.

Then the *predict phase* takes place and the particles are updated with data coming from odometry, adding a noise component, according to the following equation

$$x_i(t+\Delta t) = f(x_i(t)) + u(s(t)) + n_i(t) \tag{3.1}$$

with

- $f : \Omega \to \Omega$ Free Evolution function
- $u : \mathscr{S} \to \Omega$ Forced Evolution function
- $x_i(t)$ State Function of the i particle over time
- $s(t)$ Driving Signal
- $n_i(t)$ Evolution Noise (tried Gaussian and Uniform Distribution assumption)

As a project start, the $s(t)$ Driving Signal has been realized as a function associating to a certain (t_1, t_2) time interval a specific θ_1 steering wheel angle.

The detailed construction of the $s(t)$ Driving Signal will be explained later.

Problems

In the context of indoor self localization addressed with the particle filterm using the mean for position estimation can lead to an important estimation problem: while all the particles lie in allowed regionson the map, their mean could fall in a not allowed region [52].

Essentially then *the mean in general is not a closed operation* because of the underlying map structure.

Some workarounds have been proposed to address this problem, the most immediate is to project the not allowed computed mean position to the closer allowed position (assuming there is just one).

Obviously this workaround leads to a sub-optimal estimation and it requires a more detailed stratgey in case the closer positions are more than one, moreover the estimated trajectory can show discontinuities because of these sudden projections.

In addition, another more general problem regarding the mean arises: the mean is an optimal estimator only for unimodal distribution but there are no guarantees about the shape of the posterior PDF.

Solution

In the present context, the following approach has been used.

Once the Posterior Position PDF has been estimated, at the end of each particle filter iteration, the following analysis is performed.

First the $\Omega^{(s)}$ free space is quantized, by means of an empirically defined tessellation strategy, so define the $T = \{\omega_{i,j}\}_{i=1,...,N;j=1,...,M}$ tessellation for which it holds

$$\omega_{i_1,j_1} \cap \omega_{i_2,j_2} = \emptyset \quad i_1 \neq i_2, j_1 \neq j_2 \qquad \bigcup_{\substack{i=1,...,N \\ j=1,...,M}} \omega_{i,j} = \Omega^{(s)}$$

Then the sampled PDF is mapped in this tessellation so to generate $f : T \to \mathbb{R}$ that associates to each tile the number of particles (possibly weighted) falling inside.

The T domain is analyzed: if it is convex then the mean operation is closed hence it returns surely a valid position estimate.

If it is not convex there is no guarantee the mean is closed hence a different strategy is used.

The number of modes in f is computed: this is equivalent to solve a maximization problem on this discrete function. If f is unimodal the unique mode is used as the position estimate, while if f is multimodal the mode related to the higher probability is used as estimate.

This kind of solution has provided satisfying results but further improvements are possible and some ideas will be presented in section 3.3.5.

Simulation Path Planning A *simulation path planner* has been developed to define the simulated vehicle behaviour in the simulated environment.

Essentially this path planner needs two points

- the estimated present position
- the goal position

Then it computes a trajectory connecting this points that is compliant with a set of vincula: vehicle related and environment related.

Considering that in the simulated environment context the vehicle model is a simple holonomic one, the trajectory essentially consists of on site rotations and movements. Considering the real vehicle type that has been used for the real AGV prototype development, it has been necessary to switch to a tricycle model hence developing a different path planner version.

The *Simulation Path Planner* needs the user to input a $\{g_i\}$ *sequence of goal points* on the map that the vehicle is required to reach following the order. A sequence point is considered reached if at some time the following conditions holds

$$d(\hat{P}(t), g_i) < \varepsilon$$

with

- $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^+$ the euclidean distance measure on the plane
- $P(t)$ the real position of the vehicle (groundtruth)
- $\hat{P}(t)$ the estimated position of the vehicle
- g_i goal point
- ε distance threshold

Essentially this condition requires the simulated vehicle to get close enough, according to a certain threshold, to a goal point regardless of the vehicle orientation, in order to consider it reached.

Once the particle filter has finished an iteration, an estimate for the position on the map is provided.

This position estimate is used by the *simulated path planner*: a trajectory connecting the *estimated present position* and the *goal position* is computed then it is transformed in a Driving Signal.

Essentially given a trajectory, it is subdivided in a set of $\{l_i\}_{i=1,\dots,N_l}$ *length intervals* and to each of these intervals a certain steering angle is associated, according to the following equation

$$s(l) = \theta_i \quad l \in (l_i, l_{i+1}) \tag{3.2}$$

where θ_i represents the steering wheel angle associated to the (l_i, l_{i+1}) length interval.

Essentially the *driving signal* is a function that associates to the covered distance a steering angle.

In the present implementation, the trajectory is recomputed at each iteration using the estimated position.

Data Analyzer

The *Data Analyzer* has been developed in MATLAB because of the features provided by this language in data manipulation.

The block of data logged by the *Environment Simulator* is loaded and shown, it containes the following elements

1. $\{g_i\}$ Sequence of goal points
2. $P(t)$ Real Position Function, the groundtruth
3. $\hat{P}(t)$ Estimated Position Function

The following *quality indicators* are computed

1. $r(t) = P(t) - \hat{P}(t)$ Position Error
2. $s(t)$ Shortest Path Trajectory
3. $d_{ps}(t) = P(t) - s(t)$ Shortest Path Distance

The *Position Error* 1 represents the difference between the real position and the estimated one: the smaller this value over time the more precise is the self localization.

The *Shortest Path Trajectory* is a set contiguous segments representing the shortest possible path connecting the sequence of goal points. It is considered as a sort of ideal path because it is the cheapest one in term of battery and time consumption.

The *Shortest Path Distance* 3 represents the difference between the real trajectory and the ideal one.

Detailed Motion Simulator

The *Detailed Motion Simulator* has been developed to separately study the motion computed by the simulated path planner using the *tricycle vehicle model*, that will be used in the real AGV development, with the real vehicle measures in order to estimate the *vehicle space occupation* during its motion.

3.3.4 Simulator Milestone

The milestone consisted in the development of the 3 software modules described.

The *Environment Simulator* has been developed as a stand-alone PYHTON module responsible to replicate the real world situation of the AGV moving around the environment and performing self localization, while trying to accomplish a set of tasks essentially consisting of following a certain sequence of goal points.

This simulator provides a simple interactive GUI showing the online vehicle simulated motion in the simulated environment (see Fig. 3.2) but the main purpose is to log all the generated data in a specific infrastructure, that in the present implementation consists of files, to be later throughly analyzed by the *Data Analyzer*.

In the context of this simulator it is possible to setup important parameters like

- the environment layout
- the different kind of noises to be added to signals in the simulation (mostly Gaussian)

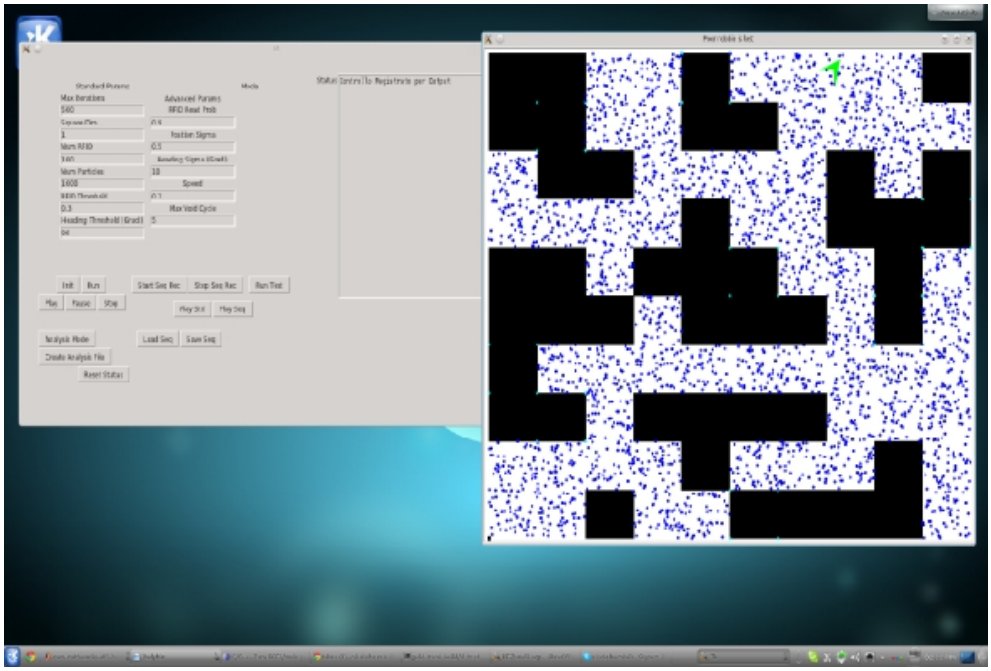


Figura 3.2: Example of simulated environment composed of two blocks: free space block containing RFID Tags and occupied block

- RFID related params like
  - RFID Reading Distance
  - RFID Reading Success Ratio
  - RFID Density on the smart floor

The *Data Analyzer* is a MATLAB script able to load and process the log files realized by the *Environment Simulator*.

In figure 3.3 it is possible to see a typical result of this module.

The image on the left represents the layout of the environment: the white blocks represent walls and the blue dots the RFID position on the ground.

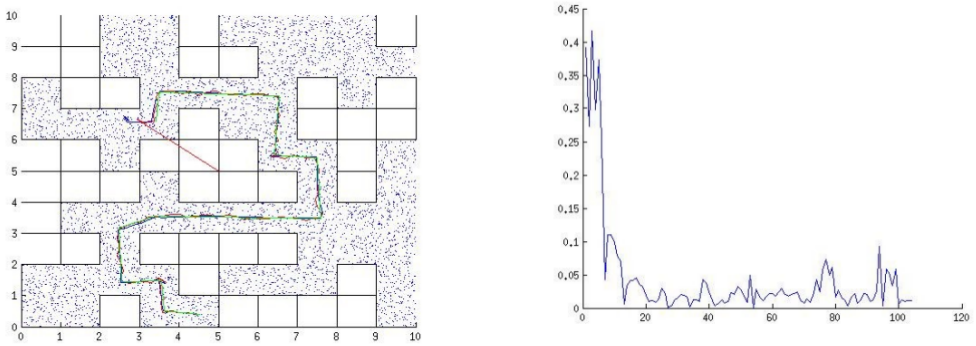


Figura 3.3: Example of Data Analyzer Result regarding a simulation.

The green line represents the $s(t)$ shortest path connecting the control points, the blue line represents $P(t)$ the real trajectory followed by the vehicle during the motion and the red line repsents $\hat{P}(t)$ the position estimation during the motion.

The image on the right represents the $d_{ps}(t)$ shortest path distance function defined as the distance between the optimal trajetory and the real trajetory.

Performing simulations with different values of the *reading distance* parameter, the following results have been obtained

Considering the results (representatives samples of all the simulations performed) shown in figures 3.4, 3.5, 3.6, 3.7 it is possible to observe the error functions to decrease with the reading distance and then to increase again. This phenomenon seems to signal a sort of *trade off* between *frequent readings*, facilitated by a high reading distance, and *precise readings*, facilitated by a low reading distance.

The *Detailed Motion Analyzer* is a MATLAB script able to process a *command sequence* sent to the vehicle and to visualize the vehicle motion using a more precise vehicle model, the tricycle model, and vehicle occupation during the motion.

In Figure 3.8 it is possible to observe a typical result.

The *command sequence* processed by this module is not related to the previously described simulation path plannet but to the path planner designed for the real AGV

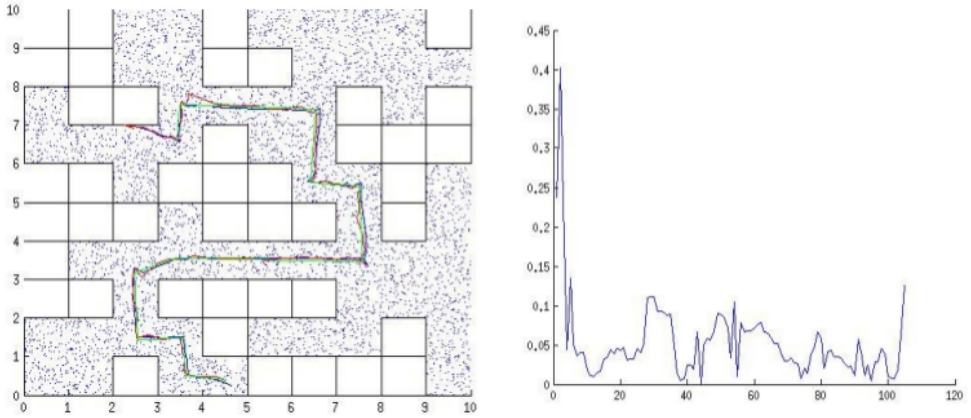


Figura 3.4: Example of Data Analyzer Result regarding a simulation.

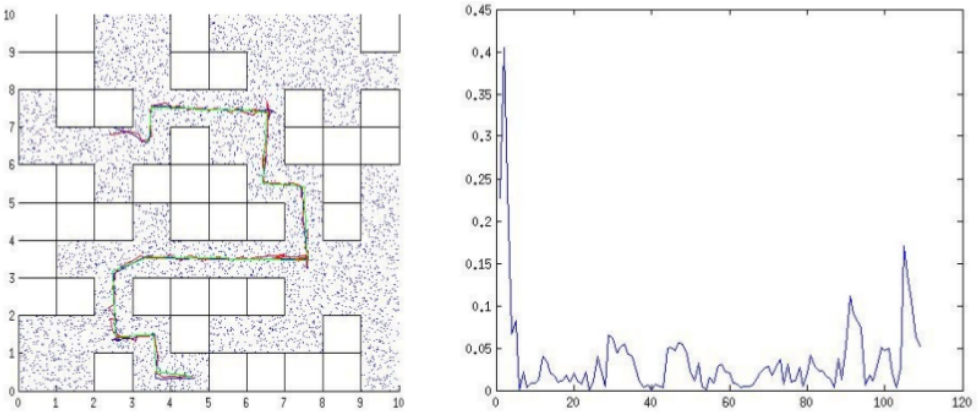


Figura 3.5: Example of Data Analyzer Result regarding a simulation.

using the tricycle vehicle model.

It generates a command sequence starting from the spline representing the vehicle desired trajectory and solving an inverse kinematics problem. A more detailed

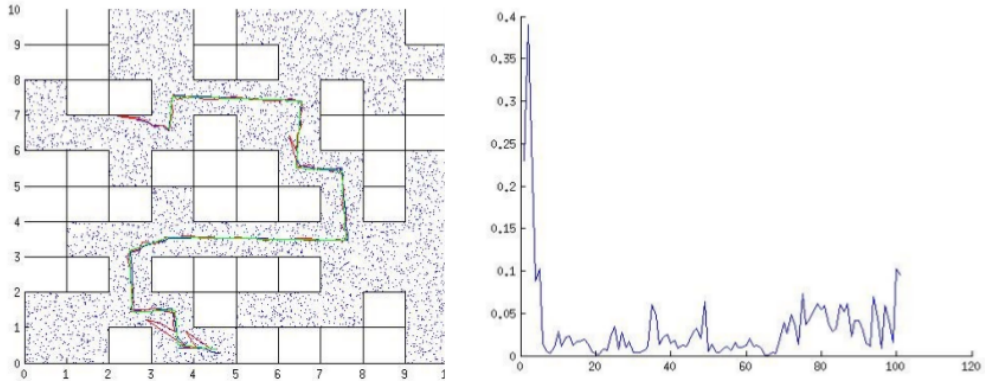


Figura 3.6: Example of Data Analyzer Result regarding a simulation.

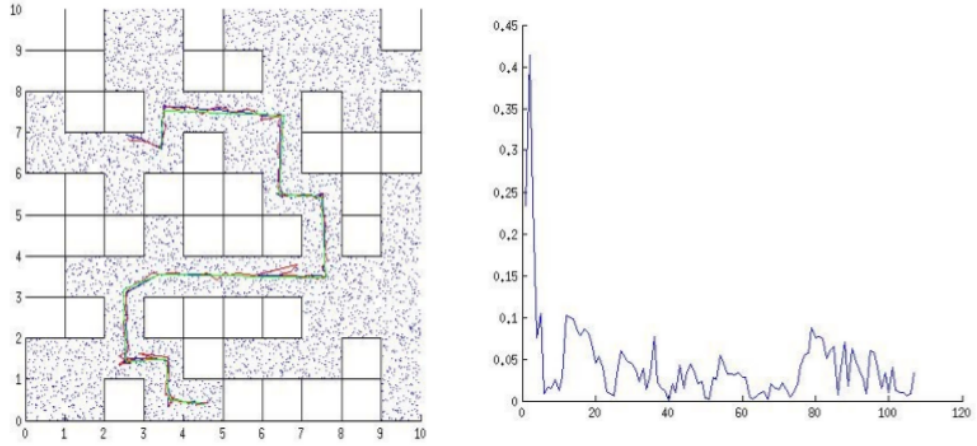


Figura 3.7: Example of Data Analyzer Result regarding a simulation.

description about this module will be provided in the AGV development section 3.4.3.

The sequence is passed to this simulator and the *vehicle occupation during the motion* is estimated. Applying this information on the map it is possible to estimate

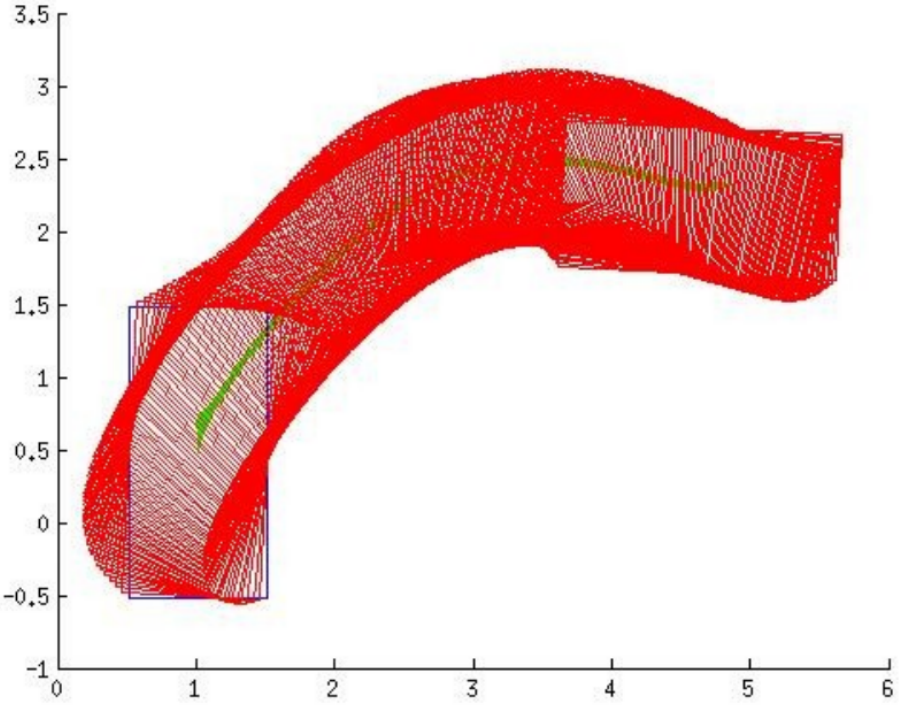


Figura 3.8: Example of Detailed Motion Analyzer.

if the desired motion is compliant with environmental vincula.

3.3.5 Further Developments

Some improvements for the next development step are here presented.

Particle Filter

The particle filter based *self localization module* implemented presented can deal with non convex domains (e.g. corners) but it can be further improved to better deal with multimodal PDFs.

With multimodal PDFs, taking the mode with the highest value is reasonable but in real situations it can lead to wrong results especially when the second minimum has a probability value very close to the first one.

However despite the simplicity of the proposed solution, the system showed good performance in the experiments carried out. The reason is probably related to the nature of the RFID measurement adopted: when it is available, a unique ID associated to a unique position is provided hence generating a very narrow particle distribution around the spatial point associated to the observed RFID Tag.

Models improvements

The quality of the simulated results depend significantly on the quality of the models in the simulator. The following models could be further improved

1. Vehicle model
2. RFID reading model

The *vehicle model* 1 could be developed on the basis of the *vehicle project* and performing some specific measurements on the real vehicle.

The *RFID reading model* 2 could be improved performing accurate physical measurements regarding the reading process also taking into account the effect of a set of solution parameters like different positions for the readers, the depth of the RFID Tag in the smart floor, ...

3.4 The AGV

In this section, the AGV development is presented.

First a general overview about the goals is provided as well as the main problems presentation. Then the main elements of a proposed solution are outlined and details and observations about its development are provided. Finally the milestone is described and comments about future developments are provided.

3.4.1 Problem Definition

The main goal of this phase consisted in the development of a working AGV prototype able to to perform the following tasks

- safely navigate a certain environment (known map) being able to rely just on the odometry for short distances
- being able to perform an *emergency stop* in case getting too close to walls, machineries or persons
- being able to autonomously pitchfork certain basins located in specific points and to carry them to different locations then releasing them there in well defined configurations

The AGV development involves dealing with additional problems compared to the simulated environment ones.

3.4.2 Proposed Solution

In order to adequately address the previously defined requirements, the navigation problem has been addressed considering two types of navigation

1. the *long distance navigation* and
2. the *short distance navigation*

The *long distance navigation* 1 is aimed at covering a long distance path in an industrial/warehouse environment. The main goal of this approach is to get *close enough* and with *right orientation* to a certain destination location so to be able to switch to a higher precision navigation strategy.

During this kind of movement, a lower precision with respect to the other kind of nagivation 2 is acceptable and in general it is possible to manage an excessive error accumulation by means of a path replanning.

The *short distance navigation* 2 is amied at covering a short distance path while achieving a higher precision (compared to the above defiend strategy) necesseray to perform an operation like pitchforking.

In this work, the *short distance navigation* 2 has been implemented exploiting computer vision information as a feedback signal while pitchforking to achieve higher accuracy during the motion.

The development roadmap has been defined as follows

1. The standard fork lift has been equipped with all devices to transform it in an AGV
2. The software architecture for the system is defined
3. Long Distance Navigation Module Development
4. Long Distance Navigation Module Test
5. Short Distance Navigation Module Development
6. Short Distance Navigation Module Test

The step 1 consists of the necessary adaptations to transform a standard fork lift in a vehicle able to perform some tasks with a certain level of autonomy, possibly an AGV.

The step 2 consists of the *software architecture* definition for the whole system including the vehicle and extrenal elements.

The step 3 regards the development of the algorithm and the complete software module to perform long distance navigation.

The step 4 consists of performing quality tests about the system performance related to movement for dozens of meters. Specific tasks have been defined and measurements have been taken in different conditions.

The step 5 regards the development of the algorithm and the complete software module to perform short distance navigation as well as the pitchforking task.

The step 6 consists of performing quality tests about the system performance related to the pitchforking activity.

Appunto

3.4.3 Development of the AGV

Vehicle Setup

Regarding the ***vehicle setup step*** 1 first the components list is presented then the system setup is explained.

Components list The following components list has been used to perform the adaptation

1. A standard fork lift
2. A PLC
3. A Modbus Module for the PLC
4. A SNDH encoder
5. An Industrial PC
6. Industrial RFID Readers
7. Industrial RFID Tags
8. Two couples of stereo cameras
9. Two SICK laser scanners
10. Two Emergency Stop buttons

The *PLC* 2 is an Omron PLC CP1L-EM40D. It has been programmed in LADDER LOGIC and some functions have been developed in STRUCTURED TEXT. The OMRON CX PROGRAMMER has been used to manage the PLC configuration and development.

The *Modbus PLC Module* 3, the *RFID Readers* 6 and the *RFID Tags* 7 are TURCK solutions.

The *RFID Reader* 6 is a TN-Q80-H1147 RFID Antenna and it is shown in Figure 3.9.

Figura 3.9: *RFID Antenna.*

The *encoder* 4 has been connected to the fork lift steering wheel to perform odometrical measurements.

The Modbus Module has been used to allow for the communication between the PLC and the Industrial PC. The used protocol has been the MODBUS TCP with PC acting as a master and the PLC acting as slave.

All the sensors, like the wheel encoder and RFID antenna, have been connected to the PLC and their values were available for reading as Modbus memory locations.

The *two stereo couples* 8 are GUPPY PRO cameras. The stereo couple looking at the forks is composed of color cameras while the other stereo couple is composed of grey tones cameras.

This choice has been made because the developed solution requires performing a *markers recognition* task and color information, in addition to the shape one, leads to very good performance.

In figure 3.10 it is possible to see the fork lift before and after the adaptation process.

Figura 3.10: *On the left, fork lift before the adaptation. On the right, the fork lift adapted to become an AGV.*

Notes

Collaboration with local SME The AGV adaptation has been realized in collaboration with a local SME company which provided the fork lift and a suitable warehouse-like test space.

Laser Scanners In the present development step, the SICK laser scanners have been employed only for safety reasons, they did not provide any information used to perform self localazation.

RFID employed The RFID number actually used for the development of the milestone presented in this work has been minimal due to the company purpose to keep the realization cost low in this initial test phase.

RFID Tags has been placed only on the starting and destination positions.

Setup Let's consider as the AGV back the side with the forks.

The RFID Readers have been assembled on the bottom of the AGV so to minimize their distance with respect to the ground.

The lasers have been assembled close to the AGV front corner so to cover

- the AGV front space up to a distance of about 40 cm
- both the AGV sides up to a distance equal to the whole forks length

in order to assure a *secure long distance navigation*.

The two stereo couples have been installed on the fork list top part, one looking forward and the other looking backward.

In figure 3.11 it is possible to see the internal structure of the AGV computation box containing both the PLC and the Industrial PC.

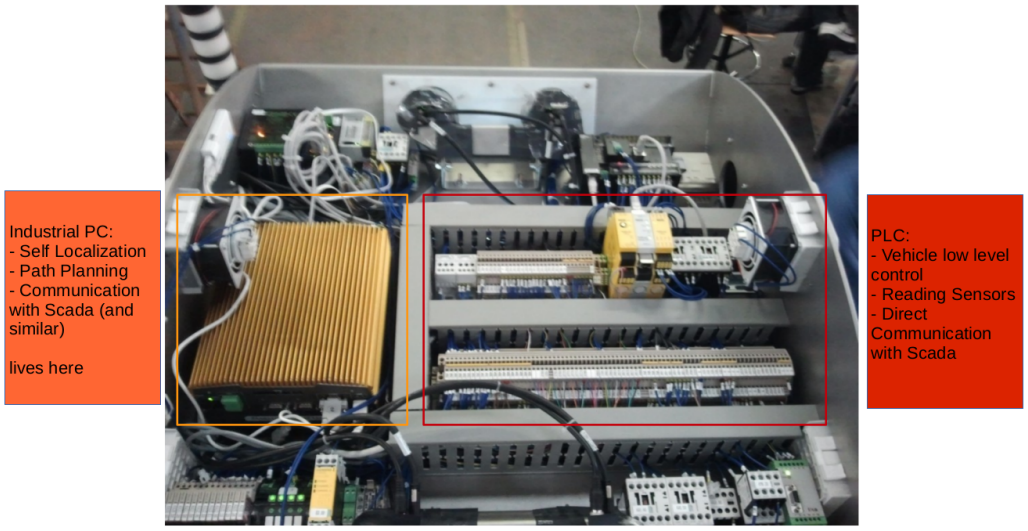


Figura 3.11: *AGV Internal. PLC and Industrial PC.*

Regarding the ***Software Architecture Definition*** 2 the global structure is presented. Essentially it consists of two macro models

- the vehicle macro module
- the SCADA system macro module

The SCADA system macro module identifies a set of plugin that can be developed for the local SCADA system to allow for the vehicle communication. Its importance regards the fact that the *industrial context* is a *supervised context*, this means that a *global monitoring and coordinating system* exists, unlike contexts like the automotive one where this is kind of facility is at the moment not present, hence it should be natural for any machine operating in this environment to be able to interface with such a system.

The vehicle macro module includes the following modules

1. self localization
2. path planner
3. task control
4. mission control

The *self localization* module 1 is responsible to compute a localization estimate. Essentially it relies on the implementation of Data Fusion Algorithms able to fuse information in a Bayesian fashion coming from *prediction*, so depending on the vehicle model and odometry data, and *measurements*, so in the context of the present work coming from RFID readings and Computer Vision.

The *path planner* module 2 is responsible to define a trajectory connecting the present vehicle position and a certain arrival position. In the context of the present work different strategies have been followed.

The *task control* module 3 is responsible to carry out, iteration by iteration, a certain task. Moreover this module performs a sort of *quality control* regarding the ongoing task. Firthermore it is possible to define policies to slow down, speed up and interrupt the task according to its quality level.

The *mission control* module 4 is responsible to break a complex mission in a set of easier missions, identified as tasks.

In the context of the present work, tasks can be categorized as follows

1. point to point long distance motion
2. pitchforking
3. objects identification

The *point to point motion* task 1 consists essentially in moving from the present point to a certain destination point also specifying the orientation hence performing a *long distance motion*.

This problem has been solved as follows

- generating a cubic spline considering the present position and orientation and the desired final position and orientation
- solving an ***Inverse Kinematics*** problem (as described in section 2.2) in order to compute the *command sequence*

The *pitchforking* task 2 consists in correctly positioning the forks in the tank housing and then moving them upward to pick it up and moving them downward to release it both in a general position and in well defined machine related positions hence performing a *short distance motion*.

The *object identification* task 3 consists of exploiting the computer vision information to identify object exposing *special markers* easily recognizable by means of computer vision pattern matching.

Actually another category of tasks exist, the *autonomous tasks* category whose elements do not need to be explicitly executed because they are continuously ongoing. These tasks are the following ones

1. security check task
2. self localization task

The *security check* task 1 performs all the security checks regarding data coming from the laser scanners. Considering the criticity level for this operation, the related

software has been implemented at PLC level, hence in a context independent from the Industrial PC.

The *self localization* task 2 is performed by the *self localization* software module 1 in the context of the Industrial PC because it is a non critical although very important task.

An example of complex mission could be: *pick up the tank and bring it to X machine*.

The *mission control* module 4 breaks it in the following tasks list

1. Locate the tank in the world (actually it is not an elementary task but a submission)
2. Move close to the element with an orientation suitable for pitchforking
3. Verify the element is the right one
4. Pitchfork element
5. Move close to the final machine

Long Distance Navigation Development

The ***long distance navigation*** 3 is realized by the *path planner* module 2.

The theory presented in section 2.2 has been exploited.

Given the vehicle starting position (known as a result of the self localization module) and an arrival position, depending on the *mission control* module 4 complex task decompisition, the *path planning* module 2 defines a *trajectory* according to a certain *policies list* and then solves an *inverse kinematics problem* to compute the *driving function* for the vehicle.

Policies for trajectory computation Policies list

1. compliancy with the vehicle vincula
2. compliancy with the environmental vincula

3. compliancy with the mission vincula

The *compliancy with the vehicle vincula* 1 regards all the vincula related to the vehicle in itself and its dynamic capabilities. Examples of this kind of vincula are the maximum steering angle, the maximum vehicle speed, ...

A particularly important vinculus belonging to this category is the *vehicle breaking capability* that has to be carefully analyzed for obvious safety reasons.

The *compliancy with the environmental vincula* 2 regards all the vincula related to the environment in itself, like walls, as well as the vincula related to the vehicle occupation of the environmental space.

The *compliancy with the mission vincula* 3 regards all the mission related vincula, like the need to approach to a certain location with a specific orientation because the following task requires it, for example to carry out a pitchforking operation.

To address the last two types of problem, the *Detailed Motion Analyzer* described in section 3.3.3 and section 3.3.4 has been developed.

Trajectory computation In the context of the present thesis, the trajectory is computed using cubic splines.

When a new trajectory needs to be computed, the *path planner* module 2 receives the following data

1. Starting position and orientation
2. Arrival position and orientation
3. Non mandatory additional vincula. For example: minimum and maximum speed.

The *Starting position and orientation* 1 are obtained querying the *self localization* module 1. In case it is failing or in case of explicit overriding, this kind of information could be obtained querying the SCADA system which is supposed to have a global knowledge of the environment.

The *Final position and orientation* 2 is typically obtained by the *mission control* module 4 but it can also be explicitely defined by means of a direct command from the SCADA.

Further *non mandatory additional vincula* 3 like minimum and maximum speed limit during the trajectory can condition the maximum steering angle (which could be speed dependent, for some kind of vehicles) hence also the trajectory shape.

In the context of the present implementation, this kind of constraints can be defined only on the whole spline, a more fine-grained control is considered for further developments (see section 3.4.5).

Hence Trajectory Planning is as follows

Algorithm 4 Trajectory Planning

Known $x \in \mathbb{R}^2, \dot{x} \in \mathbb{R}^2$ Initial Position and Orientation
Input $y \in \mathbb{R}^2, \dot{y} \in \mathbb{R}^2$ Final Position and Orientation
for $i = 1$ to 2 **do** ▷ Cycle over the 2 dimensions of the problem
Compute a cubic spline considering $(x_i, \dot{x}_i, y_i, \dot{y}_i)$
end for
Verify the spline is compatible with all the vincula
If it is not compatible signal it and wait or halt (TBD)
If it is compatible compute $\phi(t)$ driving function ▷ It is computed solving the Inverse Kinematics Problem

3.4.4 AGV Milestone

The developed system has been presented to an important sectorial fair in Nurimberg.

The system has been autonomously moving for about 8 hours a day for 3 days in total, with just occasional stops due to battery recharging.

Some missions have been pre-defined consisting of

- moving to a certain position and coming back
- moving to a certain position, picking up a tank, moving to another position and dropping it there
- safe mission halting and initial position recovery

It attracted the attention of some local professionals.

3.4.5 Further Developments

The company we have collaborated with, has provided a positive feedback on the project and demonstrated interest in proceeding with the development.

Some suggested improvements are the following ones

1. Computer Vision based Self Localization
2. smart floor development resulting in the improvement of the self localization module
3. improve integration with the SCADA system

The realization of a *computer vision based self localization* 1 solution could rely on specific markers placed inside the warehouse environment. Each marker would consist of a unique identifier and an observation of a set of them at a certain time could be an important source of information to perform indoor self localization.

The *smart floor* 2 based localization project has already been described in this chapter.

The *increasing SCADA integration* 3 depends on the need to improve the global plant intelligence.

Capitolo 4

Automotive Context

4.1 Brief Overview

The main technology for self localization in the automotive environment is GPS however it can not be really considered a *silver bullet* because some major issues exist.

First of all the GPS signal is not always available because concrete and steel buildings shield it (hence for example it does not work in tunnels) and second it is in general a quite noisy (at least when considering commercial not too expensive devices), hence a Data Fusion approach involving other source of information is recommended.

In the automotive environment, two other major sources of information are suited for this kind of task: laser scanners and vision.

In the present work, computer vision performed by means of cameras is invetsigated as complemetary source of information to carry out Data Fusion with GPS.

Actually many different kind of cameras exist, considering their wavelenght, but while for certain applications like pedestrian detection the IR Cameras provide good data, in the context of the Self Localization problem the Visible Light Cameras are the best suited.

Essentially the reason is that the cameras should capture the most unique and stable information possible related to the scene hence their focus should be on the background, which conveys more stable information in the above mentioned wavelenght range.

4.2 Automotive Project

The second part of this thesis aims at describing a project focused on approaching the self localization problem in automotive environment aiming at the implementation of a SLAM (Simultaneous Localization And Mapping) system.

The SLAM problem has already been introduced in 2.3.

The goal in this development phase regards approaching the *map building* problem.

As it has been previously introduced, different mapping techniques exist according to the desired abstraction level. The higher the abstraction, the more invariant and semantically meaningful is the content, the higher the possibility to build *unique place identifiers*.

The idea is hence to start building a *metric map* with raw data sensors and then increase the abstraction level aiming at building a *topological map*.

To perform map building in this early development phase, the (groundtruth) GPS signal as well as the vehicle odometry have been exploited but certainly in future development the *visual odometry* implementation will substitute the GPS.

This project is divided in two phases

- the Dataset development
- the Real Vehicle development

In the Dataset development, algorithms have been developed and tested using common automotive related datasets.

In the Real Vehicle development, data has been acquired using the Deeva Vislab vehicle and online experiments have been designed and presented.

Appunto

4.3 Dataset development

During this early algorithm development phase a common dataset has been used: the Kitti Dataset [53], [54], [55].

It is a widely used dataset in the automotive environment for applications like stereo, optical flow, visual odometry, 3D object detection and 3D tracking.

It provides the following data

- high-resolution color and grayscale image sequences
- groundtruth consisting of Velodyne laser scanner and a high quality GPS signal

The dataset presents images belonging to rural, urban and highway scenarios, taken driving around Karlsruhe (DE). Pedestrians and cars are also present in the images.

4.3.1 Problem Definition

The *Map Building* task is the first problem approached.

In the context of the present work a roadmap is defined aimed at building different kind of environment maps for SLAM purpose (see 2.3.4).

The first kind of map that has been approached from an applicative point of view in the present work is a *feature map*.

Building this kind of map requires solving the following non trivial the problems

1. selecting the best points, filtering away all the points related to non stable objects which provide information that should not be included in a map
2. computing the most accurate 3D position estimation possible for each point
3. storing this information in a data structure optimized for fast and reliable data search

The *best point selection* 1 problem is ascribable to other known computer vision problems.

A good point for a SLAM mapping application is a point related to some element of the background and not to the foreground ones: it should belong to the most invariant part of the context like for example buildings. Moving objects in front of the vehicle camera, like other vehicles, pedestrians, ... could provide wrong feature points which would introduce noise in the map. Methods for backrgound / foreground discrimination could be applied to reduce this kind of mapping noise.

The Kitti Dataset images contain pedestrians and cars making the mapping task more difficult with respect to the same task performed on a Dataset without these foreground moving objects.

The *3D position estimation* 2 for each point is a widely studied topic in computer vision and essentially it is solved performing a *bundle adjustment* computation on a set of observations of the same point, as descrived in section 2.7.

The *fast searching on a large dataset for a feature point* 3 is a field of study recently very active.

Large image datasets, consisting of over 1 million items, have recently become available and the old linear feature comparison algorithms show poor scalability hence on the standard commercial hardware satisfactory performance have not been observed raising the need for new approaches. Regarding this topic, literature has been presented in section 2.9.

4.3.2 Proposed Solution

In the context of the present work, the first goal concerned the construction of a *feature map*. Starting from the raw frames sequence provided by the dataset, *stereo from motion* is performed and a Cloud of 3D Points is computed. Each point has an associated descriptor depending on the local image content.

This kind of map shows little abstraction: there is no concept like *place* and hence neither like *place fingerprint* yet but it is basis the build a *topological map* in future steps.

Due to the lack of *unique place identifiers* in this map, in order to achieve good performance in self localization a reliable vehicle model and good odometrical data

are needed so to be able to compute a good *Prior* about the vehicle position in the above mentioned feature map.

The *predicted measurement* (depending on the prior-related position in the map) can then provide good (unambiguous) data to be compared with the *observed measurement* hence defining the *measurement error*.

Essentially the performance of this strategy depends on the assumption that the *prediction model* is good enough to avoid the *map ambiguity* to significantly degrade the performance.

The development roadmap has been defined as follows

1. The Kitti Dataset needs adaptations to be used with the Vislab Framework GOLD

2. An application that extracts good feature points from the image sequence and associates them with the camera rototranslation matrix is developed

3. An application that loads the data generated by the feature extraction app to build a map consisting of a cloud of 3D points is developed

4. The generated map is verified

The step 1 consists of the necessary adaptations to load the dataset sequence of images and the dataset groundtruth data in the Vislab Framework for processing.

The step 2 consists of the development of a preprocessing module that extracts good feature points from the each image in the sequence and associates them with the camera rototranslation matrix in a data structure.

The step 3 consists of the development of a processing module that loads the above mentioned data structure and for each feature point a triangulation problem is solved so to compute an approximation for the 3D coordinates of that feature point hence building a *3D points cloud map*.

The step 4 consists of the verification of the map by means of a visual inspection.

Once the map has been built, the localization algorithm needs to be defined.

Appunto

4.3.3 Development on Kitti Dataset

Dataset Adaptation

Regarding the ***dataset adaptation*** 1 first a brief overview of the Vislab Framework GOLD for Computer Vision processing is provided.

The framework has been developed with the goal to record and load its own sequences hence specific proprietary conventions (file names, data structures, ...) are adopted, however it is possible to load third party sequences if the necessary adjustments are realized.

In general, the images format is not an issue because GOLD is able to load all the common formats.

Usually the above mentioned adaptations regard:

- the creation of configuration files
- renaming the image files

Feature Extraction Application

Regarding the ***feature extraction application development*** 2 first the application structure is presented. It is composed of the following elements

1. Main Cycle
2. Feature Extractor
3. Feature Quality Check
4. Position Vector Extractor
5. Data Saving

The *main cycle* 1 of the application is focused on sequentially loading all the images in the sequence so to perform all of the above described operations on them.

In the *feature extractor* 2 module the FAST detector and BRIEF256/BRIEF512 descriptors have been used, essentially for performance related reasons.

The *feature quality check* 3 is responsible for selection the best features for the following computation. A stability criterion is applied.

Since a certain feature appears in a frame, it is tracked searching for the same feature inside a *search window* whose dimension are empirically estimated, centered on its previous position, across the subsequent frames. If the feature is detected for a number of frames superior to a certain minimum threshold, it is considered a *stable feature* otherwise it is discarded.

Furthermore at each iteration in a certain frame, features with the same descriptor are selected and only the most stable one is kept while the others are discarded.

The *position vector extractor* 4 is responsible for extracting the camera roto-translation matrix related to the current frame. In this application, this data is simply got directly from the groundtruth but in following development steps it could be estimated by the odometrical data.

The *data saving module* 5 is responsible to store each feature that passed the quality test in the current frame with rototranslation matrix corresponding to the current frame.

Note on feature quality test The feature quality test needs information coming from frames different from the considered one.

Let's consider $\{G_i\}_{i=1,\dots,N}$ the *feature objects array* in memory related to the features observed up to the i frame. Its structure is $G_i = \{g_{i,j}\}_{j=1,\dots,N_f(i)}$ where $N_f(i)$ represents the number of features in the i frame, hence G_i contains $N_f(i)$ objects $g_{i,j} = (f_{i,j}, n_{i,j})$ which is an association between a $f_{i,j}$ feature point and a $n_{i,j}$ counter that represents the number of consecutive frame the feature has been observed for.

New features will be revealed and added to G_i with $n_j = 0$ while disappearing features will be revealed observing the condition $n_{i,j} = n_{i+1,j}$.

These features will be removed by the $\{G_i\}$ array and if $n_{i,j} > n_{th}$ they are considered stable feature so their whole story will be stored in another data structure for subsequent processing, otherwise they are discared due to instability.

Stable features array structure is

$$\{(f_i, \{p_{i,j}\})_{j=1,\dots,N_s(i)}\}_{i=1,\dots,N_{tot}} \tag{4.1}$$

with

- N_{tot} Total Number of Features revealed in the whole sequence
- f_i Feature Descriptor
- $N_s(i)$ Number of consecutive frames the feature has been observed
- $\{p_{i,j}\}_{j=1,\ldots,N_s(i)}$ Set of Rototranslation Matrix each related to a different observation of the feature point

The Feature Extraction and Pre Map Data Structure building algorithm is presented in algorithm 5

Algorithm 5 Pre Map Data Structure Building Algorithm - Main Cycle

$\{f_j\}_{j=1,\ldots,N_f(I_i),i} = F_{ext}(I_i)$ ▷ Feature Extraction from the current image
All the *action_flag* in the *features object array* are unset ▷ This flag signals which feature has been modified in this cycle
for $i = 1$ to N **do** ▷ Cycle over all the feature observed
If the considered feature is already present in the *features object array* the counter is updated and a *action_flag* is set ▷ Feature counter update
If the considered feature is not present in the *features object array* it gets added and *action_flag* is set ▷ New Feature
end for
for $i = 1$ to M **do** ▷ Cycle over all the featurs in the *feature object array* with the *feature_flag* unset namely the disappeared feature
If the feature counter is greater than a certain threshold, the feature information are saved otherwise it is discared
end for

The notation for this algorithm is the following

- F_{ext} Feature Extraction function
- I_i the i frame

- $N_f(I_i)$ Function that return the number of features detected in a certain image
- $\{f_j\}_{j=1,\ldots,N_f(i),i}$ Number of features detected in the i frame

Map generation application

Regarding the ***map generation application*** 3 first the application structure is presented. It is composed of the following elements

1. Load Data
2. Bundle Adjustment
3. Computation of the back projection error
4. Point Quality Check
5. Map blocks subdivision
6. Data Saving

The *load data* 1 module is responsible to load the data presented in 4.1 resulting from the Feature Extraction Application.

The *bundle adjustment* 2 module is responsible to setup a multiview triangulation problem as described in section 2.7 for the observed feature and to solve it with a least square minimization.

The *back projection error computation* 3 module is responsible to compute the *back projection error* which is used as a quality measure for the point.

The *point quality check* 4 module is responsible to determine the quality of each computed 3D point so to decide wether include it or not in the final map.

The criterion presently used for the quality check consists of a comparison of the back projection error with a threshold (empirical threshold). If this error is larger than this value the point is discarded and will not be used in the map.

A large back projection error indicates a poor estimation. Moreover it could be due to the relative movement of the observed feature with respect to the vehicle. It happens for example in case the extracted feature has been extracted by the image

of another car. In this situation, it is desiderable not to include that point in the map because it is not part of the background but of a foreground object.

The *map block subdivision* 5 divides the map in rectangular blocks of $\{B_i\}$ with l_x, l_z dimension.

The *map data save* 6 module is responsible to save the points that passed the quality test in the final map which consists of a cloud of 3D points, each associated with a certain descriptor.

The Map Building Algorithm is presented in algorithm 6

Algorithm 6 Map Building Algorithm

Pre Map Data Structure Loaded
for $i = 1$ to N **do** ▷ Cycle over all the features
Load all the poses it has been observed
Compute $\hat{x} \in \mathbb{R}^3$ Solving the minimization problem
Compute $\hat{\sigma}_x$ Estimated variance as a result of the backprojection error
If $\hat{\sigma}_x > \sigma_{th}$ the backprojection error is too high and the points is discarded otherwise it is added to the map
end for
$M = \{x_i, f_i\}_{i=1,\dots,N}$ the Map is created
$M = \bigcup_i B_i$ Map subdivision
M saved

The M Map consists of a cloud of N 3D points with $x_i \in \mathbb{R}^3$ the coordinates resulting from the solution of the minimization problem, each associated with a f_i binary descriptor.

Localization Algorithm

The developed version of the map is essentially a *cloud of 3d points* and each of these has an associated *binary descriptor*.

This map can be used in a two step localization process performed in a Bayesian Framework as follows

1. Prediction Step

2. Update Step

The *prediction step* 1 essentially relies on the *odometrical data* applied to a *vector model* as well as to the *visual odometry*.

The *update step* 2 compares the *observed frame* with the information in the map.

Let's point out that it is not an image to image comparison, because the map is not composed of image, although it is a feature search problem: the features extracted from the observed images are directly searched in the map data structure.

It is important to consider two important issues

1. Descriptors are not map unique identifiers for points

2. Descriptors difference

The fact that the *descriptors are not unique identifiers for map points* 1 is due to the fact that these values are computed just considering the image local feature information, the map is not involved so in a map it is not unlikely that different points in different positions have the same descriptor.

Hence considering f_q the query feature descriptor, the search should not involve all the map points: the $\tilde{P}(X)$ *predicted position* PDF is used to identify a $S \subset M$ small region in the map (empirical threshold) to perform the search on so to use the *update information* and compute the $\hat{P}(X)$ *posterior position* PDF. Considering $\{f_i\}_{i=1,\dots,n} \in S$ the set of all features identified in the above mentioned subregion, a linear search with the query feature is carried out.

At this point it is important to consider the *descriptor difference* 2 problem: essentially the observed descriptor can be different from all the candidate descriptor points. This issue could be due to the fact the keypoint is observed from a slightly different point of view or more generally to some kind of observation noise.

Because of this, an exact matching can not be carried out but a similarity distance comparison is used. Considering binary descriptors have been employed, the Hamming Distance has been chosen.

Considering the following notation

- $\tilde{P}(X)$ *predicted position PDF*
- $S(\tilde{P}(x);\theta) \subset M$ *Subset of the map* depending on $\tilde{P}(X)$ predicted position and θ some parameters (empirical rules and thresholds)
- $S(\tilde{P}(X);\theta) = \{f_j\}$ *Subset of the map* consisting og of n feature points
- W *observed keypoints*
- $\{f_i\}^{(q)}$ Query Features that is a set of N elements
- $\mathscr{F}$ Feature Space
- $d : \mathscr{F} \times \mathscr{F} \to \mathbb{R}^+$ Feature Distance
- d_{min} *minimum similarity distance* (empirical threshold)
- $L = \{i\}_j \quad j = 1,...,N' \leq N$ set indexes of associated features
- $V_L = \{p_i\}_{i \in L}, V_L \subseteq W$ set of observed points with an associated map feature
- f the function that fuses the prior estimation and the measurement to compute the posterior
- $\hat{P}(X)$ *posterior position PDF*
- $\hat{x}$ *posterior position estimate*

The localization algorithm is described in algorithm 7
In the localizatio algorithm 7 there are 3 fundamental elements

1. $S(\tilde{P}(X);\theta)$ the Predicted PDF Filter which identifies the Map Subregion
2. the *posterior prefiltering logic* which determine if a certain query feature is present in the map
3. the f *Data Fusion Function* that computes the Posterior PDF

Algorithm 7 Localization Algorithm

The $\tilde{P}(X)$ Predicted Position PDF is computed
The $S(\tilde{P}(X);\theta) \subset M$ Subset of the Map is computed
for $i = 1$ to N **do** ▷ Cycle over all the elements in $\{f_i\}^{(q)}$
$\hat{j} = \arg\min_{j=1,\dots,n} d(f_i^{(q)}, f_j)$ ▷ Look for the most similar feature
If $d(f_i^{(q)}, d_{\hat{x}}) < d_{min}$ create association otherwise reject
end for
L, V_L are created
$\hat{P}(X) = f(S(\tilde{P}(x);\theta), L)$ the Posterior PDF is computed
$\hat{x} = E[\hat{P}(X)]$ the Posterior Estimation is performed

In the present implementation, the *Map subregion identification* 1 relies on the $M = \{B_i\}$ *map blocks subdivision* previously described.

The $f_i^{(q)}$ *query feature points* are projected in the image space using the estimated *vehicle camera matrix* (depending on the estimated *vehicle position*) and only the resulting visible features are considered for matching.

Regarding the *Posterior Pose Estimation* the following strategy has been adopted.

Once the V_L *observed keypoints set* has been defined, a *3D pose estimation problem* is solved using Direct Linear Transform (DLT) that estimates all the 12 parameters in the projection matrix using 6 points hence $\tilde{C}_{\{p_j\}_{j=1,\dots,6}}$ *estimated projection matrix* is defined, according to $\{p_j\}$ *set of random points* such as $p_j \in V_L \quad \forall j = 1,\dots,6$

The $\tilde{C}_{\{p_j\}}$ *estimated projection matrix* is used to compute $\tilde{V}_{L,\{p_j\}}$ *projection of the observable map points in the image space* so that a comparison with respect to V_L *observed keypoints* can be performed such as the following set is defined, where each element is the difference between a map point projected in the image space according to the estimated projection matrix and its related observed point

$$D = \{\tilde{p}_i - p_i\} \qquad p_i \in V_L, \tilde{p}_i \in \tilde{V}_{L,\{p_j\}} \tag{4.2}$$

Then setting a certain threshold, RANSAC is used to discriminate inliers and outliers so $R_{\{p_j\}}$ *inliers number* is used a score for $\tilde{C}_{\{p_j\}}$.

The whole process is repeated up to N_{iter} iterations.

Finally $\hat{C}$ is determined as the projection estimation with the highest score hence the highest number of computed inliers.

4.3.4 Kitti Dataset Milestone

In the present version, the map is succesfully built out of the Kitti Dataset Sequences. Various maps are shown, with different parameters setup. The considered parameters are the following ones

- N_{fs} Number of contiguous frames for considering a feature as stable
- σ_{th} Standard Deviation Threshold (in meters)
- l_x, l_z Width and length of a generic B_i map block (both in meters)

In figure 4.1 it is possible to see a frame of the computed map representing the point cloud related to a certain place.

In figure 4.2 an overview of the whole map is shown using BRIEF256.

In figure 4.3 another overview of the whole map is provided but using BRIEF512. The quality has clearly improved.

Localization has also been performed on map represented in figure 4.3 and synthetic results are shown in Figure 4.4. In world coordinates, the blue crosses represent the groundtruth and the green crosses represent the estimated positions.

In this early development phase, the same mapping sequence has been used.

As it was expected, a good match between the groundtruth set and the estimated positions set is observed.

Regarding the quality of the *pose estimation* the Figure 4.5 shows the quality score, hence the number of the inliers, associated to each pose estimation matrix along the whole sequence. The higher the value, the better the pose estimation quality. The Figure 4.6 shows the percentage of the inliers.

A clear performance drop is observed in the regions between spike and spike. This strange behaviour needs to be further investigated.

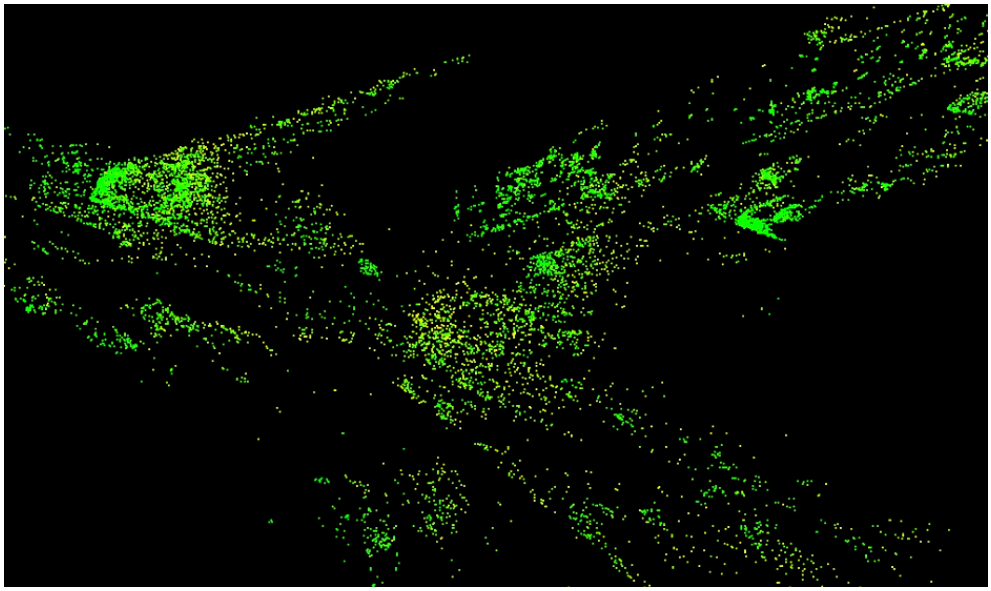


Figura 4.1: SLAM map. Detailed location. BRIEF256.

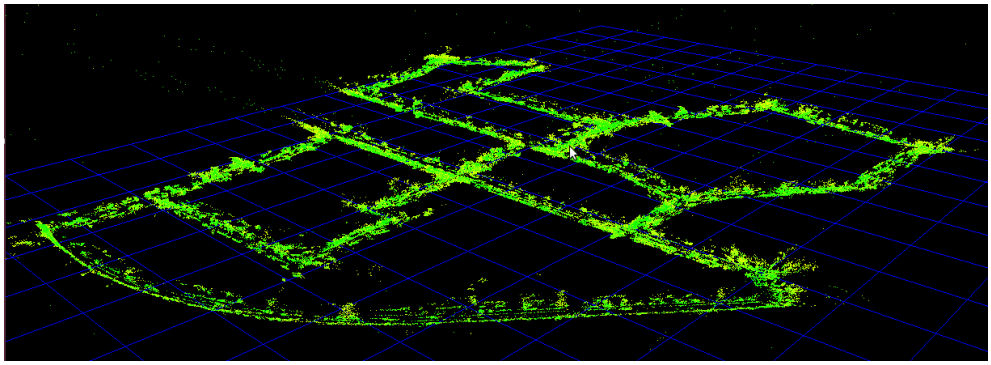


Figura 4.2: Whole Map. BRIEF256. $N_{fs} = 4, \sigma_{th} = 0.5, l_x = 30, l_z = 30$

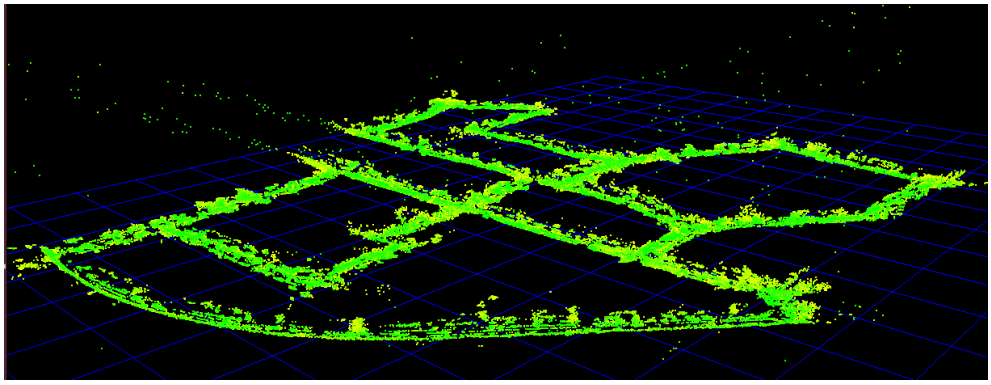


Figura 4.3: Whole Map. BRIEF512. $N_{fs} = 4, \sigma_{th} = 0.5, l_x = 30, l_z = 30$

In Figure 4.7 the estimated map is shown. The quality of this results is quite satisfactory.

In Figure 4.8, 4.9, 4.10, 4.11, 4.12, 4.13 the errors related to roll, pitch, yaw, x, y, z estimation are provided as a difference between the groundtruth and estimated value.

4.3.5 Further Developments

Presently a map has been generated out of a dataset sequence as the result of an application developed on the Vislab Computer Vision Framework GOLD, a mature tool aimed at developing computer vision applications especially for the automotive environment.

Regarding the ***Mapping Task*** the following improvements are considered for close future development

1. Visual Odometry development
2. Background / Foreground discrimination methods development
3. Loop Closure detection

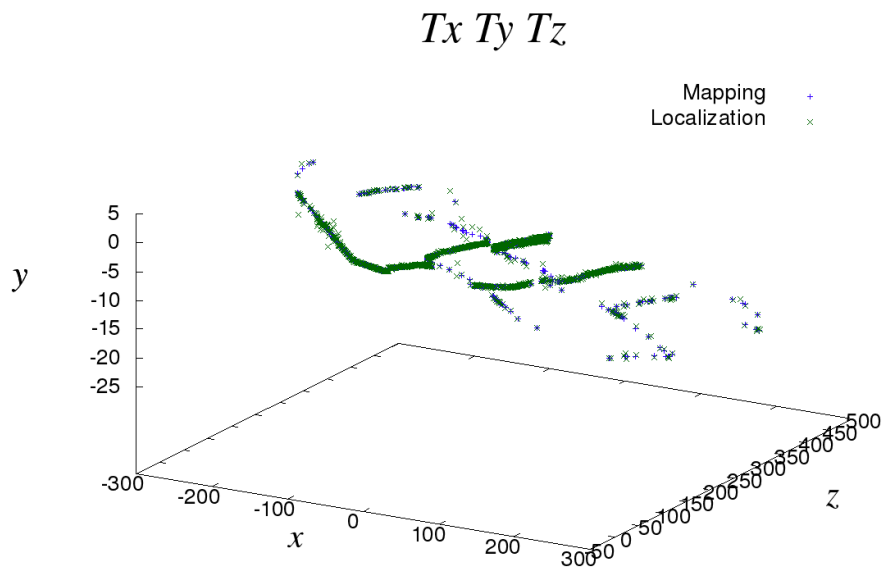


Figura 4.4: Localization performance. The (x, y, z) represent world coordinates. Blue crosses are the groundtruth and green crosses the estimated positions.

The *Visual Odometry* 1 module or better a module able to perform data fusion between wheel odometry and visual odometry will be used instead of the groundtruth data for the vehicle position.

The *Background / Foreground discrimination* 2 module will be used to drop features on foreground objects so to improve the quality of the map.

The *Loop Closure Detection* 3 is known by literature to be a very important step in the SLAM development roadmap.

Regarding the ***Localization Task*** an early prototype has been developed.

Considering the developed map, the following strategy has been used:

1. First the *prior* estimate needs to be computed

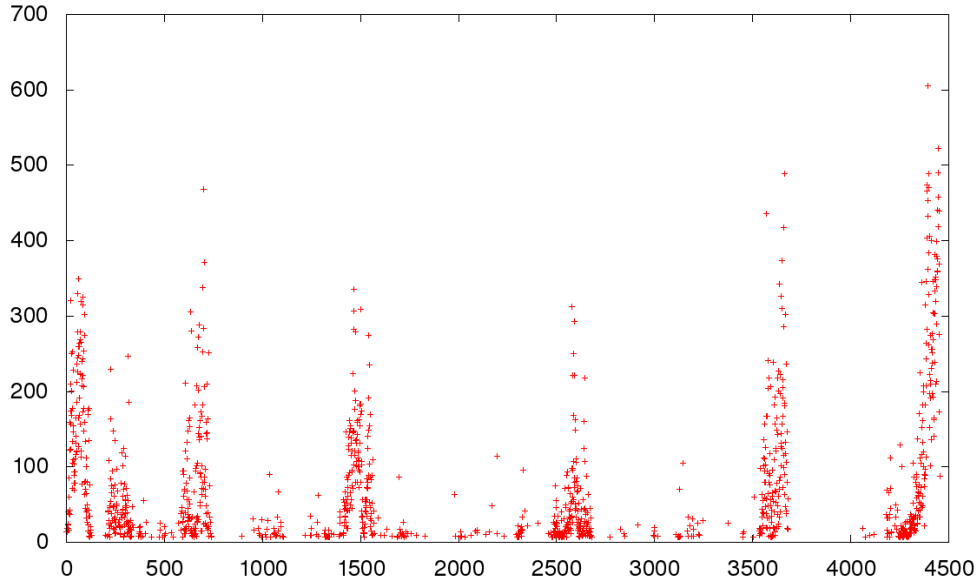


Figura 4.5: Absolute number of inliers computed for the pose estimation matrix along the whole sequence. On the x-axis the number of the sequence frame is shown. On the y-axis the number of inliers related to the best pose matrix estimation is shown.

2. The *prior* is exploited to identify a *submap* centered around it

3. In the *submap* a *comparison* between the features related to the observed scene and the map features is performed

4. Finally position is estimated

The *prior computation* 1 is a necessary step to perform a *reduction* of the problem dimension, focusing only on a specific submap.

The *submap computation* 2 is carried out exploiting the *prior* and it allows to identify a *subset of map points* which are close to the the prior estimated positions.

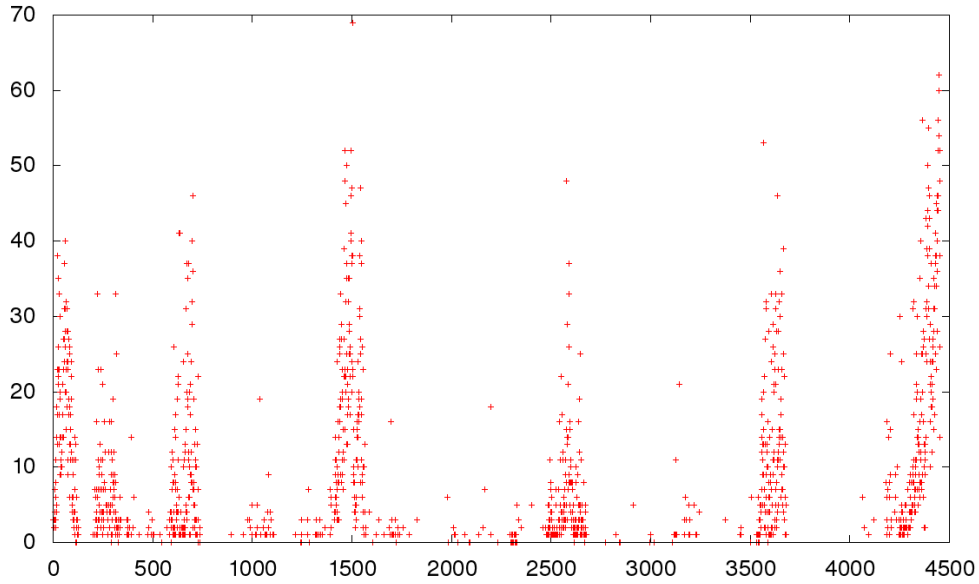


Figura 4.6: Percentage of inliers computed for the pose estimation matrix along the whole sequence. On the x-axis the number of the sequence frame is shown. On the y-axis the number of inliers related to the best pose matrix estimation is shown.

Different policies regarding the submap estimation should be analyzed, in particular a parametric relation involving the quality of the prior estimation should be investigated: the better the quality of the prior, the smaller the submap can be.

The identified subset of points is thoroughly analyzed by means of a *feature comparison* 3 strategy, involving the *observed features set* and the *features in the submap set*.

Different strategies should be considered to perform the best association possible between observed points and map points, taking into consideration the effect of second minima.

Finally the mapping and the localization processes should be carried out simultaneously and online in order to realize a real SLAM system.

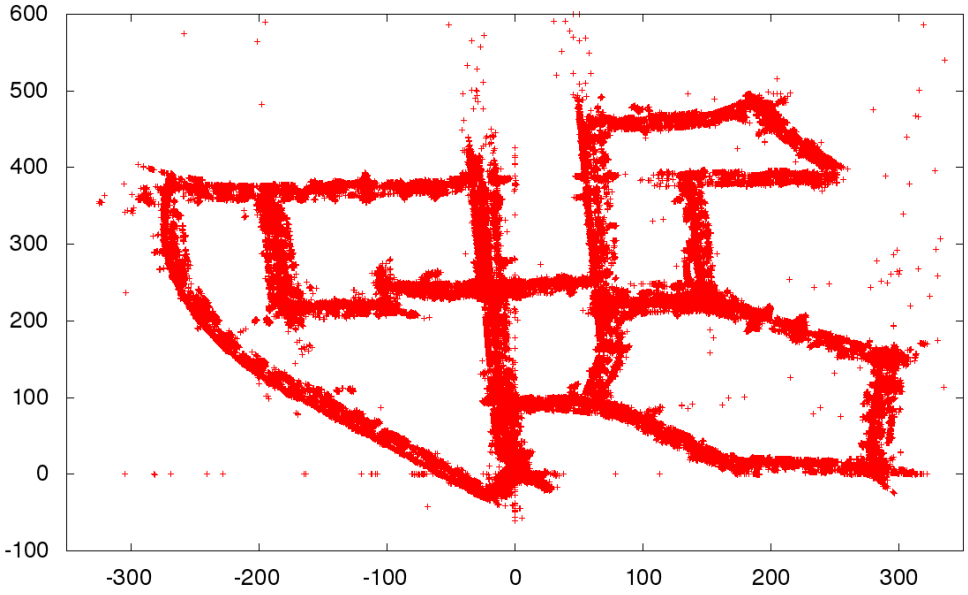


Figura 4.7: Percentage of inliers computed for the pose estimation matrix along the whole sequence. On the x-axis the number of the sequence frame is shown. On the y-axis the number of inliers related to the best pose matrix estimation is shown.

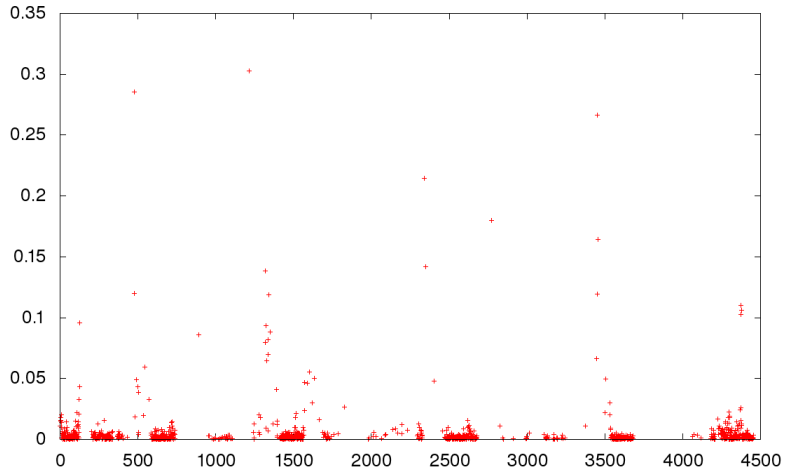


Figura 4.8: Roll Error. On the x-axis the number of the sequence frame is shown. On the y-axis the roll error is shown.

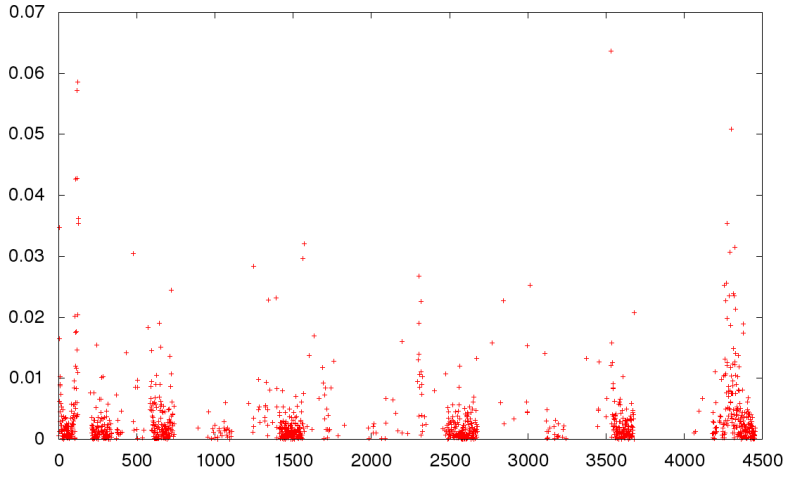


Figura 4.9: Pitch Error. On the x-axis the number of the sequence frame is shown. On the y-axis the pitch error is shown.

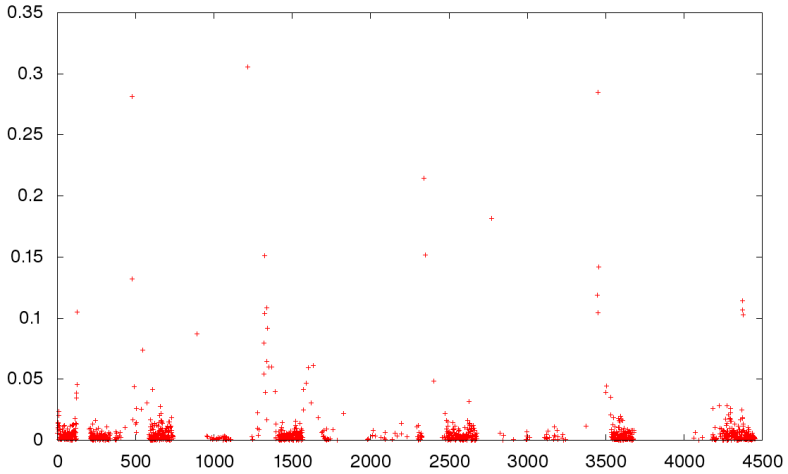


Figura 4.10: Yaw Error. On the x-axis the number of the sequence frame is shown. On the y-axis the yaw error is shown.

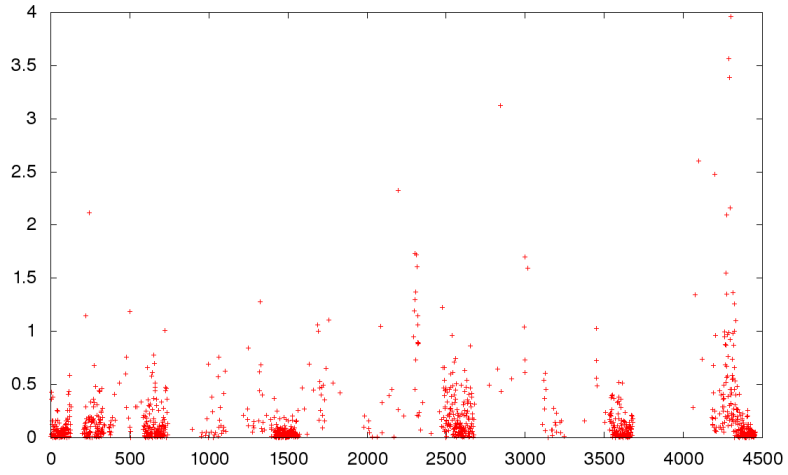


Figura 4.11: X Error. On the x-axis the number of the sequence frame is shown. On the y-axis the x error is shown.

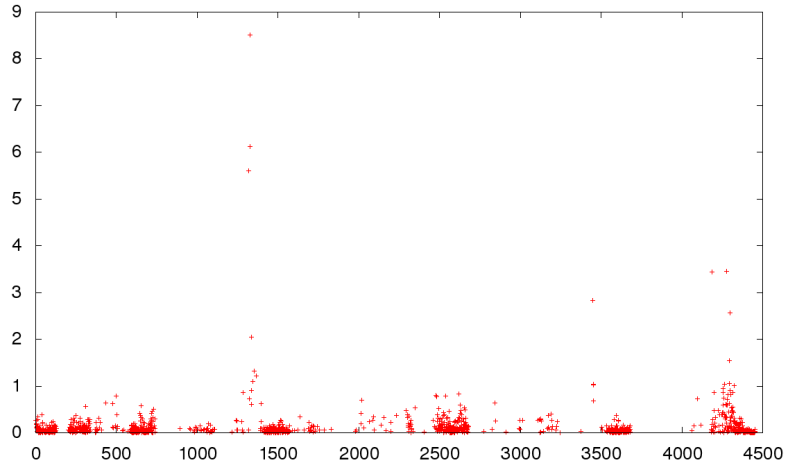


Figura 4.12: Y Error. On the x-axis the number of the sequence frame is shown. On the y-axis the y error is shown.

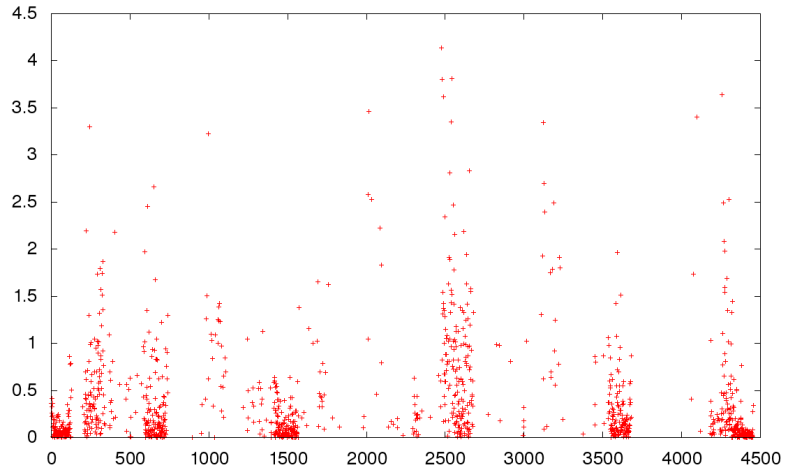


Figura 4.13: Z Error. On the x-axis the number of the sequence frame is shown. On the y-axis the z error is shown.

Capitolo 5

Conclusions

This thesis investigated the problem of self localization with both from a theoretical and an applicative perspetive for the industrial and the automotive environment.

As a consequence of this work the following results have been obtained: three project milestones, some accepted and submitted peer review articles, other articles still under development and the present manuscript.

First the localization problem has been approached in a controlled environment, the industrial/warehouse one.

At the beginning a simulator has been developed to focus on specific issues then the problem has been approached in real world with an applicative perspective. A working AGV prototype, able to perform self localization and autonomous driving under certain conditions has been developed for the industrial/warehouse environment and it has been presented to an important sectorial fair.

Then the localization problem has been considered in the wider context of the SLAM for the automotive environment.

Working with a widely known automotive dataset, a software application able to build a map using the dataset information has been developed and a solution to perform localization has been outlined.

The development of these two projects required a multidisciplinary approach. First the problem of self localization, especially in the SLAM context, has been ap-

proached from a theoretical point of view making use of mathematical formalism to thoroughly define the problem and to outline solution strategies.

The problems of prototyping and developing working software solutions are computer science / programming problems, requiring specific technical knowledge like

- C/C++, PYTHON development knowledge
- GOLD (the C/C++ proprietary Vislab Framework) knowledge

During the AGV development phase, some electric/electronic engineering knowledge was required in particular for the machine setup.

In the second part of the thesis, the roadmap to implement a SLAM system in the automotive environment has been defined.

Working prototypes for the mapping and the localization tasks separately have been realized and results regarding the application on a widely known dataset have been shown.

Specific further developments suggestions have been proposed at the end of each project.

Bibliografia

[1] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, 2(3):24–33, 2003.

[2] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[3] Mário AT Figueiredo and José MN Leitao. Unsupervised image restoration and edge location using compound gauss-markov random fields and the mdl principle. *Image Processing, IEEE Transactions on*, 6(8):1089–1102, 1997.

[4] S Derin Babacan, Rafael Molina, and Aggelos K Katsaggelos. Generalized gaussian markov random field image restoration using variational distribution approximation. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1265–1268. IEEE, 2008.

[5] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[6] Nhat Vu and BS Manjunath. Shape prior segmentation of multiple objects with graph cuts. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[7] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Computer VisionâĂŤECCV 2002*, pages 82–96. Springer, 2002.

[8] George Vogiatzis, Philip HS Torr, and Roberto Cipolla. Multi-view stereo via volumetric graph-cuts. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 391–398. IEEE, 2005.

[9] Jorge Torres-Solis, Tiago H Falk, and Tom Chau. A review of indoor localization technologies: towards navigational assistance for topographical disorientation. *Ambient Intelligence*, pages 51–84, 2010.

[10] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[11] L Gracia and J Tornero. Kinematic control of wheeled mobile robots. *Latin American applied research*, 38(1):7, 2008.

[12] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[13] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Real-time monocular slam: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE, 2010.

[14] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[15] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[16] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.

[17] Solomon Eyal Shimony. Finding maps for belief networks is np-hard. *Artificial Intelligence*, 68(2):399–410, 1994.

[18] Elias Dahlhaus, David S Johnson, Christos H Papadimitriou, Paul D Seymour, and Mihalis Yannakakis. The complexity of multiway cuts. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251. ACM, 1992.

[19] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986.

[20] DM Greig, BT Porteous, and Allan H Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 271–279, 1989.

[21] Yair Weiss and William T Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *Information Theory, IEEE Transactions on*, 47(2):736–744, 2001.

[22] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[23] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):147–159, 2004.

[24] Jian Sun and Marshall F Tappen. Learning non-local range markov random field for image restoration. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2745–2752. IEEE, 2011.

[25] Prasanna K Sahoo, SAKC Soltani, and Andrew KC Wong. A survey of thresholding techniques. *Computer vision, graphics, and image processing*, 41(2):233–260, 1988.

[26] Sang Uk Lee, Seok Yoon Chung, and Rae Hong Park. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52(2):171–190, 1990.

[27] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[28] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International journal of computer vision*, 22(1):61–79, 1997.

[29] Xavier Bresson, Selim Esedoáÿąlu, Pierre Vandergheynst, Jean-Philippe Thiran, and Stanley Osher. Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and vision*, 28(2):151–167, 2007.

[30] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, 1990.

[31] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1101–1113, 1993.

[32] Paul B Chou and Christopher M Brown. The theory and practice of bayesian image labeling. *International Journal of Computer Vision*, 4(3):185–210, 1990.

[33] George R Cross and Anil K Jain. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983.

[34] Haluk Derin and Howard Elliott. Modeling and segmentation of noisy and textured images using gibbs random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):39–55, 1987.

[35] Sridhar Lakshmanan and Haluk Derin. Simultaneous parameter estimation and segmentation of gibbs random fields using simulated annealing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(8):799–813, 1989.

[36] BS Manjunath and Rama Chellappa. Unsupervised texture segmentation using markov random field models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):478–482, 1991.

[37] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[38] Pedro F Felzenszwalb. Representation and detection of deformable shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(2):208–220, 2005.

[39] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[40] Yixin Chen and James Z Wang. Image categorization by learning and reasoning with regions. *The Journal of Machine Learning Research*, 5:913–939, 2004.

[41] Özge Öztimur Karadağ and Fatoş T Yarman Vural. Image segmentation by fusion of low level and domain specific information via markov random fields. *Pattern Recognition Letters*, 2014.

[42] Sébastien Roy and Ingemar J Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Computer Vision, 1998. Sixth International Conference on*, pages 492–499. IEEE, 1998.

[43] Sébastien Roy. Stereo without epipolar lines: A maximum-flow formulation. *International Journal of Computer Vision*, 34(2-3):147–161, 1999.

[44] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[45] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 26–33. IEEE, 2003.

[46] George Vogiatzis, Philip Torr, Steven M Seitz, and Roberto Cipolla. Reconstructing relief surfaces. In *In BMVC*. Citeseer, 2004.

[47] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[48] Duncan P Robertson and Roberto Cipolla. An image-based system for urban navigation. In *BMVC*, pages 1–10, 2004.

[49] Hao Shao, Tomáš Svoboda, Tinne Tuytelaars, and Luc Van Gool. Hpat indexing for fast object/scene recognition based on local appearance. In *Image and Video Retrieval*, pages 71–80. Springer, 2003.

[50] Wei Zhang and Jana Kosecka. Image based localization in urban environments. In *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pages 33–40. IEEE, 2006.

[51] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168. IEEE, 2006.

[52] Robert Piche and Mike Koivisto. A method to enforce map constraints in a particle filter's position estimate. In *Positioning, Navigation and Communication (WPNC), 2014 11th Workshop on*, pages 1–4. IEEE, 2014.

[53] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[54] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[55] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013.