SLAM and Vision-based Humanoid Navigation

Olivier Stasse

▶ **To cite this version:**

Olivier Stasse. SLAM and Vision-based Humanoid Navigation. Humanoid Robotics: A Reference, pp.1739-1761, 2018, 978-94-007-6047-9. 10.1007/978-94-007-6046-2_59 . hal-01674512

HAL Id: hal-01674512
https://hal.science/hal-01674512

Submitted on 3 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SLAM and Vision-based Humanoid Navigation

Olivier STASSE, LAAS-CNRS

1 Motivations

In order for humanoid robots to evolve autonomously in a complex environment, they have to perceive it, build an appropriate representation, localize in it and decide which motion to realize. The relationship between the environment and the robot is rather complex as some parts are obstacles to avoid, other possible support for locomotion, or objects to manipulate. The affordance with the objects and the environment may result in quite complex motions ranging from bi-manual manipulation to whole-body motion generation. In this chapter, we will introduce tools to realize vision-based humanoid navigation. The general structure of such a system is depicted in Fig. 1. It classically represents the perception-action loop where, based on the sensor signals, a number of informations are extracted. The informations are used to localize the robot and build a representation of the environment. This process is the subject of the second paragraph. Finally a motion is planned and sent to the robot control system. The third paragraph describes several approaches to implement visual navigation in the context of humanoid robotics.

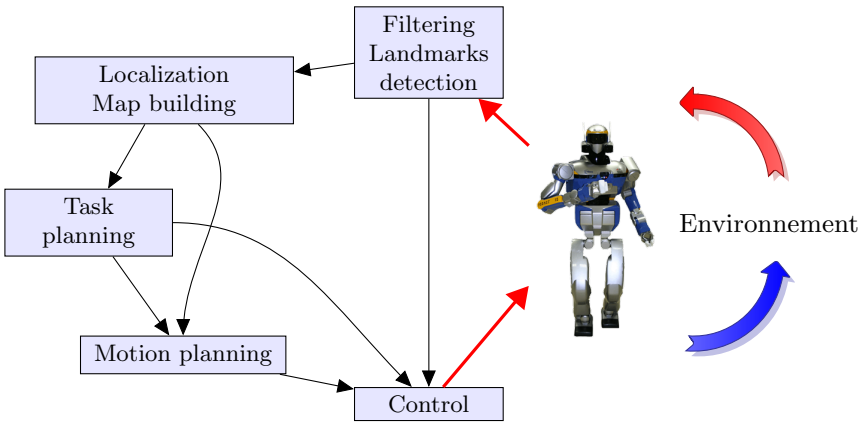


Fig. 1: Control and data flow for visual navigation.

1.1 Examples

An early example of vision based navigation for a humanoid robot is the work presented by Seara [49]. In this study the humanoid robot Johnnie is actively controlling its gaze system to follow a path and avoid obstacles. Visual search is

also an example of high level behavior implying active vision. In [1], ASIMO finds a previously learned object by choosing the next motion which will maximize the expectancy of having the object in the field of view. Real-time footstep planning was also realized by Chestnutt and al. on ASIMO using a remote vision system [4]. Another example of similar high level behavior was implemented on the humanoid robot HRP-2 looking for an object inside an unknown environment [46]. In 2013, Nishiwaki [37] demonstrated how to make HRP-2 able to build in real-time a map of the environment using a Hokuyo laser range finder and plan foot-steps into it. The most recent example and probably one of the most challenging was obtained in the context of the Darpa Robotics Challenge (DRC) where robots were teleoperated to accomplish various tasks such as walking on rough terrain, remove debris and climb a ladder []. During the DRC trials in December 2013, most of the teams relied on a shared autonomy scheme where a human in the loop was asserting the quality of the planning.

1.2 Formalism

Behavior realization can be formalized as follows. Considering a robot with a command vector $\mathbf{u}$ of size n, with a vector of information related to its internal parameters and the environment $\mathbf{v} \in \mathbb{R}^m$. For a given behavior let us assume that there exists a function $f(\mathbf{u}, \mathbf{v}, t) : \mathbb{R}^{n \times m+1} \rightarrow [0, 1]$, which equal to 0 when the behavior is realized. The problem amounts to find a trajectory $\mathbf{u}(t)$ such that

Behavior (B)

$$\min f(\mathbf{u}(t), \mathbf{v}(t)) \tag{1}$$

$$\mathbf{g}(\mathbf{u}(t), \mathbf{v}(t)) < 0 \tag{2}$$

$$\mathbf{h}(\mathbf{u}(t), \mathbf{v}(t)) = 0 \tag{3}$$

where $\mathbf{g}$ are unilateral constraints and $\mathbf{h}$ are bilateral constraints.

A common approach in robotics is to build a function $\hat{f}$ which is an approximation of f based on a model of the current state $\mathbf{v}(t)$, and an estimation of the current state of the robot in this environment. A central issue is to find the appropriate formulation of $\hat{f}$ and $\mathbf{v}(t)$ to solve B efficiently.

1.3 Constraints

The constraints that are to be taken into account are:

- **Dynamics of the system**: the control and the dynamics of the robot have to be consistent
- **Externals forces**: with the ground in particular and the environment in general are necessary to guarantee balance. For complex motion the dynamics is strongly linked with the robot dynamics.
- **Avoiding joint limits, and torque limits**: this is necessary to avoid burning motors and breaking the system.

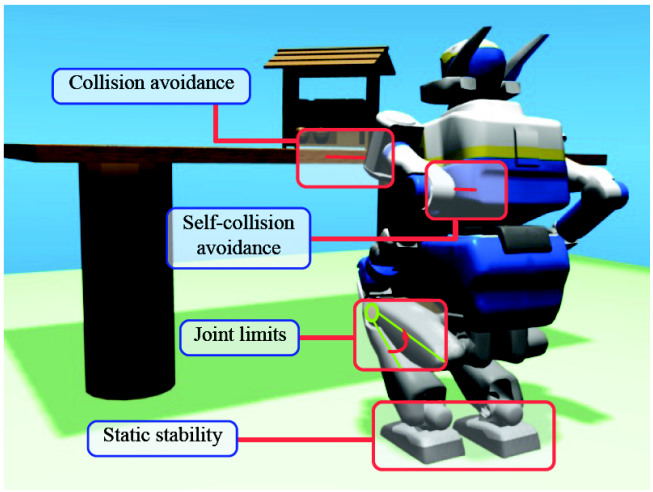


Fig. 2: Inspecting an object and the necessary constraints that need to be taken into account when generating a motion

– **Avoiding collision of the robot**: with itself and with the environment. This constraint is difficult because it is creating a nonlinear coupling between the control variables. Moreover, it depends on the geometrical representation of the environment and the robot. Strictly convex representation are usually preferred to avoid discontinuity in the gradient and the associated control law [50]. Sweeping-spheres are usually a practical representation with a low computational cost.

To realize a behavior B, while taking into account this constraint, a navigation system has to find a sequence of controls taking into account the reconstructed environment and the constraints of the robot. Humanoid robots have however a very large space of possible actions, especially when considering multi-contacts locomotion. Exploring this space is therefore particularly time-consuming and complex due to the nonlinear interactions between the self-collision constraints and the control variables. A practical solution is to consider a discrete subset of feasible motions $\mathcal{A}$ and to find a sequence of actions which realize the behavior without colliding with the environment.

1.4 Actions

A compact way of representing an action is to consider a set of active controllers continuously running with their reference trajectories. Any removal or insertion of a controller, or discontinuous change in the reference trajectories can be seen as a new action. Ideally, one would like to be able to partition the space of possible actions based upon the available controllers and their accessible sets.

This is very difficult to approximate for simple system and seems to be out of reach for humanoid robots [32]. One of the practical problem comes from the complexity of building an accurate model of the robot to reflect the real hardware limitation (current limits, torques limit, compliancy of the hardware). It is therefore more simple to define a priori a subset of actions well tested on the robot and to plan directly in this subset. This approach was first formulated by Chessnutt and al. in [5]. Note that there is a recent trend to build more advanced controllers to simplify the work of the planner. This is achieved by having more powerful controllers able to find footsteps by themselves [20], or to minimize a visual criteria [14,15]. However when collision has to be avoided, it is necessary to deal with the geometry spanned by the controllers movements. As it is difficult to represent the whole geometry that a set of controlled movements can describe, an efficient way to deal with it is to start from a feasible solution and locally deform it by solving an optimization problem [5,42].

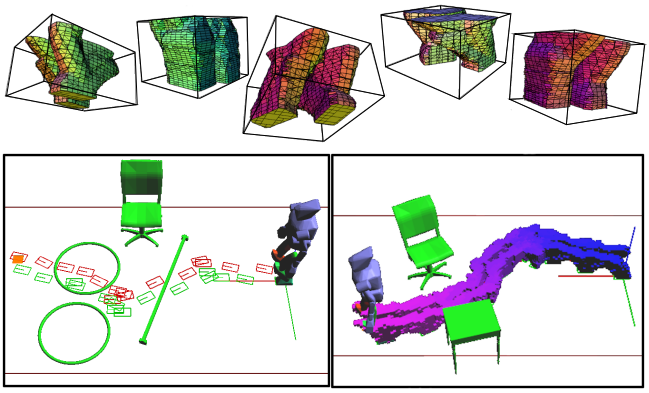


Fig. 3: 3D representations of 5 swept volumes approximations, (up) and complex planning based on a given set of motions (bottom).

1.5 Impacts

Impacts is often an issue, because they propagate along the mechanical structure of the robot creating blurred images. One simple way to cancel them out is to use the information provided by the walking pattern generator and ignore the images when the flying foot is scheduled to land. This can be done also by using the force sensor information. A more advanced way to solve the problem is to generate a motion such that the foot landing is very smooth and does not generate impact. This strategy is used in the control framework implemented on ASIMO.

1.6 Sensors

In general, laser, computer vision, RGB-D (Red Green Blue-Depth) or time-of-flight cameras provide the robot with the signal from which landmarks are computed. Humanoid robots are although subject to embeddability constraint. For this reason, only small lasers with limited time of scan are implemented on humanoid robots. RGB-D cameras offer a much large set of data, but the problem is then to have enough CPU power to handle it. GPU or massive parallel cards are usually used to solve the problem. They can be embedded inside the robot only if there is enough space, and an appropriate cooling system. It is often not the case.

The following describes more formally how the particularity of humanoid robots can be taken into account when solving the SLAM problem.

2 Simultaneous Localization and Map-building

2.1 Problem statement

Definition 1. *SLAM: Simultaneous Localization And Map-building*
For a robot this problem consists simultaneously in building a map of an unknown environment and self-localizing the robot inside it.

SLAM is a central problem when the robot has no apriori knowledge on the environment in which it has to walk. Indeed the map is used by the humanoid robot system to plan footsteps or whole body motions.

Two kinds of maps were initially considered separately: an *occupancy grid* [34] or a collection of *landmarks*. They mostly are grounded on two kinds of measurement on the environment: range-image (2D like laser-scan, or 3D like laser-scan/RBG-D images) or images provided by cameras. An occupancy grid integrates the range-image, while the landmarks are points in the environment which have a particular appearance. Since the work of Newcombe [36], visual slam system are now using the two kinds of maps together [33] and is able to scale this approach to large environment [54].

The relationship between the robot position and the measurements is given by a motion model integrating the control law computed for the robot.

Let us now define more formally the problem. By considering the formulation proposed in [11]. The state of the humanoid robot at a time step i is denoted by $\mathbf{x}_h(i)$. Here $t = i\Delta T_s$, with ΔT_s the acquisition period of the sensor s. For the sake of simplicity, a conventional linear discrete time state for the robot motion is considered and the transition equation is given by:

$$\mathbf{x}_h(i+1) = \mathbf{F}_h(i)\mathbf{x}_h(i) + \mathbf{u}_h(i+1) + \mathbf{w}_h(i+1) \tag{4}$$

where $\mathbf{F}_h(i)$ is the state transition matrix, $\mathbf{u}_h(i)$ is the vector of control inputs, and $\mathbf{w}_h(i)$ the vector of temporally uncorrelated process noise errors with zero

mean and covariance $\mathbf{R}(i)$. The position of landmark k is then denoted by $\mathbf{X}_k$. The state transition of the k-th landmark is

$$\mathbf{X}_k(i+1) = \mathbf{X}_k(i) = \mathbf{X}_k \tag{5}$$

Considering N landmarks, the vector of all landmarks is denoted

$$\mathbf{X} = [\mathbf{X}_1^T \cdots \mathbf{X}_N^T]^T \tag{6}$$

The augmented state vector containing both the state of the humanoid and the state of all landmark location is denoted by

$$\mathbf{x}(i) = [\mathbf{x}_h^T(i)\ \mathbf{X}_1^T\ \cdots\ \mathbf{X}_N^T]^T \tag{7}$$

The augmented transition model for the complete system is now:

$$\begin{bmatrix} \mathbf{x}_h(i+1) \\ \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} = \begin{bmatrix} \mathbf{F}_h(i) & 0 & \dots & 0 \\ 0 & \mathbf{I}_1 & \dots & 0 \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{I}_N \end{bmatrix} \begin{bmatrix} \mathbf{x}_h(i) \\ \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} + \begin{bmatrix} \mathbf{u}_h(i+1) \\ \mathbf{0}_1 \\ \vdots \\ \mathbf{0}_N \end{bmatrix} + \begin{bmatrix} \mathbf{w}_h(i+1) \\ \mathbf{0}_1 \\ \vdots \\ \mathbf{0}_N \end{bmatrix} \tag{8}$$

$$\mathbf{x}(i+1) = \mathbf{F}(i)\mathbf{x}(i) + \mathbf{u}(i+1) + \mathbf{w}(i+1) \tag{9}$$

where $\mathbf{I}_i$ is the identity matrix of size $dim(\mathbf{X}_i) \times dim(\mathbf{X}_i)$, and $\mathbf{0}_i$ is a vector of size $dim(\mathbf{X}_i)$ filled with zeros.

The next step is to build the relationships between the extended state given by Eq.7 and the measurements.

2.2 Observations models

General formulation As specified previously a main particularity of humanoid robots (or complex robots) is their large set of sensors: Inertial Measurement Unit (IMU) which include gyrometers and accelerometers, cameras (CCD or CMOS), encoders, RGB-D sensors, or less commonly lasers, skins. For each sensor s, the existence of an observation model is assumed. It is relating an observation with the state of the humanoid robot. A measurement of the landmark k by the sensor s at time i is denoted by $\mathbf{z}_{k,s}(i)$. Its linearization is given in the following form:

$$\begin{aligned} \mathbf{z}_{k,s}(i) &= \mathbf{H}_{k,s}\mathbf{x}(i) + \mathbf{w}_{k,s}(i) \\ &= \mathbf{H}_{\mathbf{X}_k,s}\mathbf{X}_k(i) - \mathbf{H}_{h,s}\mathbf{x}_h(i) + \mathbf{w}_{k,s}(i) \end{aligned} \tag{10}$$

where $\mathbf{H}_{k,s}$ is called the observation matrix of landmark k with sensor s. It relates the output $\mathbf{z}_{k,s}$ of sensor s to the state vector $\mathbf{x}(i)$ when observing landmark X.

In the case of a camera, let us consider the observation model that can be used.

Camera model Recently several camera models have been proposed in addition to the classical pin-hole model. For instance the omnidirectional model [48] can be found on top of some humanoid robots such as in the Robocup Humanoids League. They have the advantage to give a large view of the environment. Recently the *generalized camera model* [6] was introduced to control robots with multi camera models. However the pin-hole model is the most current one. It is found in most of the humanoid robots. Indeed because of their size CCD cameras or CMOS cameras are easier to integrate on human size and low-cost humanoid robots. The pin-hole model is described by Eq. (11)-(13) using the intrinsic parameters matrix:

$$\mathbf{K} = \begin{bmatrix} \alpha_x & \gamma & c_x \\ 0 & \alpha_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

with $\alpha_x = \lambda.m_x$ and $\alpha_y = \lambda.m_y$, λ is the focal length, m_x and m_y the scale factors of the camera pixels. γ is the skew factor of the pixel and is usually set to 0. (c_x, c_y) is the principal point and ideally $(0, 0)$. The extrinsic parameter matrix gives the rotation of the camera and the position of the camera center in the world frame. The relationship between the position of a landmark in the world $[X_l^x \ X_l^y \ X_l^z]^T$, or in the camera frame $[X_l^{x,C} \ X_l^{y,C} \ X_l^{z,C}]$ and their projection on the image plane is given by:

$$\bar{q}_l^z \begin{bmatrix} u_l \\ v_l \\ 1 \end{bmatrix} = \begin{bmatrix} \bar{q}_l^x \\ \bar{q}_l^y \\ \bar{q}_l^z \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} \ \mathbf{T} \end{bmatrix} \begin{bmatrix} X_l^x \\ X_l^y \\ X_l^z \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} X_l^{x,C} \\ X_l^{y,C} \\ X_l^{z,C} \\ 1 \end{bmatrix} \tag{12}$$

The projection function is defined by

$$p(\bar{\mathbf{q}}_l) = \begin{bmatrix} \bar{q}_l^x / \bar{q}_l^z \\ \bar{q}_l^y / \bar{q}_l^z \end{bmatrix} = \begin{bmatrix} u_l \\ v_l \end{bmatrix} = \mathbf{q}_l \tag{13}$$

with $\bar{\mathbf{q}}_l \in \mathbb{R}^3$, $\mathbf{q}_l \in \mathbb{R}^2$. Thus, with respect to the notation introduced in the previous paragraph $\mathbf{q}_l$ is the measurement of landmark $\mathbf{l}$, $\mathbf{z}_{k,\mathbf{l}} = \mathbf{q}_l$.

Camera velocity and features variation Eq. 13 is the nonlinear part of the model, it is usually linearized using the interaction matrix that links the variation of the position with the variation of the measure. Our presentation follows the presentation of De Luca [30]. For sake of simplicity, it is assumed now that the scaling factors are equal to one ($m_x = m_y = 1$), the principal point $(0, 0)$, the camera position set to the origin, then $\mathbf{q}_l = [\lambda X_l^x / X_l^z \ \lambda X_l^y / X_l^z]^T$.

$$\begin{bmatrix} \dot{u}_l \\ \dot{v}_l \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{X_l^z} & 0 & \frac{-u}{X_l^z} \\ 0 & \frac{\lambda}{X_l^z} & \frac{-v}{X_l^z} \end{bmatrix} \begin{bmatrix} \dot{X}_l^x \\ \dot{X}_l^y \\ \dot{X}_l^z \end{bmatrix} = \mathbf{J}_1(u_l, v_l, \lambda) \begin{bmatrix} \dot{X}_l^x \\ \dot{X}_l^y \\ \dot{X}_l^z \end{bmatrix} \tag{14}$$

where λ is the scale factor. It means that when only one camera is used then the map is obtained up to this scale factor. Fortunately it is possible to use the information given by the IMU and the optical flow to calibrate this scale factor [39]. This is also possible with two or more cameras using classical calibration techniques [18]. Now, the interesting point is to relate the *camera* velocity $(V, \Omega) \in \mathbb{R}^6$ to the feature variation $\mathbf{q} = [u\ v]^T$. As

$$\begin{bmatrix} \dot{X}_l^x \\ \dot{X}_l^y \\ \dot{X}_l^z \end{bmatrix} = -V - \Omega \times \begin{bmatrix} X_l^x \\ X_l^y \\ X_l^z \end{bmatrix} \tag{15}$$

then

$$\begin{bmatrix} \dot{X}_l^x \\ \dot{X}_l^y \\ \dot{X}_l^z \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 & -X_l^z & X_l^y \\ 0 & -1 & 0 & X_l^z & 0 & -X_l^y \\ 0 & 0 & -1 & -X_l^y & X_l^y & 0 \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix} = \mathbf{J}_2(\mathbf{X}_l) \begin{bmatrix} V \\ \Omega \end{bmatrix} \tag{16}$$

Therefore the following equation expresses the feature variation according to the camera velocity:

$$\begin{aligned} \begin{bmatrix} \dot{u}_l \\ \dot{v}_l \end{bmatrix} = \mathbf{J}_1\mathbf{J}_2 \begin{bmatrix} V \\ \Omega \end{bmatrix} &= \begin{bmatrix} \frac{\lambda}{X_l^z} & 0 & \frac{u_l}{X_l^z} & \frac{u_l v_l}{\lambda} & -(\lambda + \frac{u_l^2}{\lambda}) & v_l \\ 0 & -\frac{\lambda}{X_l^z} & \frac{v_l}{X_l^z} & \lambda + \frac{v_l^2}{\lambda} & -\frac{u_l v_l}{\lambda} & -u_l \end{bmatrix} \begin{bmatrix} V \\ \Omega \end{bmatrix} \\ &= \mathbf{J}_l(\mathbf{q}_l, X_l^z) \begin{bmatrix} V \\ \Omega \end{bmatrix} = \dot{\mathbf{q}}_l \end{aligned} \tag{17}$$

This relationship can be used in various manner. By tracking the points between two images, the variation $\dot{\mathbf{q}}_l$ for all the landmarks detected in the image pair can be computed. The problem is then to find the camera velocity which will best explain the landmarks displacement. It acts then as a visual odometry system. It is then a new measurement of the robot velocity.

One can also use this relationship to regulate the error between the current feature positions and a desired ones. The control system is then generating a control to minimize the error in the image plane. This is called *visual servoing*.

However more generally humanoid robots have several sensors and the set of all measurements relationships with the generalized state variables can be conveniently represented using a graphical model. As there is a causal relationship between the random variables of the problem, it is modeled as a *Bayesian Network* [3] as depicted in Fig.4.

2.3 Control

An important characteristic of the model given by Eq. 4 when considering humanoid robots is the fact that the control $\mathbf{u}$ is generally computed such that the Center-Of-Pressure follows a reference trajectory. This reference trajectory is usually defined as the center of the footsteps. When footsteps become parts of the control vector, as it happens to perform visual servoing or active vision control, the oscillations due to the transition between the left and the right foot has to be handled. This can be done either by reconstructing the feature motion [12,41], or by filtering the visual odometry [41] .

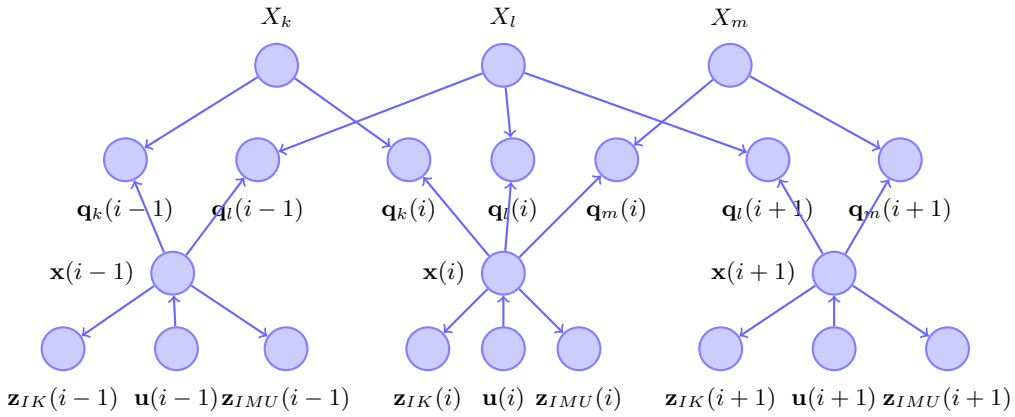


Fig. 4: Graphical model of the random variables used in SLAM for humanoid robots. $\mathbf{z}_{IK}(i)$ is the odometry provided by inverse kinematics at time i. Whereas $\mathbf{z}_{IMU}(i)$ is the measure of angular velocity and linear acceleration provided by the IMU at time i.

2.4 Estimation - Solving the SLAM problem

Once, the system collects a sequence of observations $\mathbf{Z}(i)$ until time i, solving the SLAM problem consists in computing the generalized state variable $\mathbf{x}(i)$ which has the highest probability according to $\mathbf{Z}(i)$.

SLAM

$$\mathbf{x}^*(i) = \underset{\mathbf{x}(i)}{\operatorname{argmax}}\, p(\mathbf{x}(i)|\mathbf{Z}(i)) \tag{18}$$

If all the noises are assumed to be independent and Gaussian, then it is equivalent to maximize the expectation $E[\mathbf{x}(i)|\mathbf{Z}(i)]$. However the full SLAM problem is able to retain all the constraints described by the sensor models and avoid inconsistency [52].

Full SLAM

$$\begin{aligned} \mathbf{x}(i) &= \operatorname{argmax}_{\mathbf{x}(i)} J(\mathbf{x}(i)) \\ J(\mathbf{x}(i)) &= \mathbf{x}_0^T \Omega_0 \mathbf{x}_0 + \textstyle\sum_i \mathbf{xf}^T(i)\mathbf{R}^{-1}(i)\mathbf{xf}(i) \\ &\quad + \textstyle\sum_i \sum_j \mathbf{xz}_j^T(i)\mathbf{Q}_j^{-1}(i)\mathbf{xz}_j(i) \\ \mathbf{xf}(i) &= \mathbf{x}(i) - \mathbf{f}_h(\mathbf{u}(i), \mathbf{x}(i-1)) \\ \mathbf{xz}_j(i) &= \mathbf{z}_j(i) - \mathbf{h}_j(\mathbf{x}(i)) \end{aligned} \tag{19}$$

Where $\mathbf{f}_h$ is the nonlinear function which describes the generalized state dynamics $\mathbf{x}(i)$ according to the previous state $\mathbf{x}(i-1)$ and the control $\mathbf{u}(i)$. $\mathbf{xf}(i)$ is the difference at time i between the state of the robot and the prediction of its state using its dynamic model, its previous state and the control used at time i.

$\mathbf{xz}_j(i)$ is the difference at time i between the measurement of feature j and the prediction of this feature using its model $\mathbf{h}_j$.

One way to compute this probability is to build a factor graph based on the Bayesian-Network depicted in Fig. 4. This can be done by proper variables re-ordering and elimination [25]. However if this problem is more accurate and avoid inconsistency which are usually seen with filter, the optimization problem specified by Eq.19 is computationally demanding.

Particle filter Solving the problem given by Eq. 18 can be also done by the method called Monte Carlo Localization [22], which estimates the probability of the robot's poses $\mathbf{x}(i)$ at iteration i:

$$p(\mathbf{x}(i)|\mathbf{Z}(i),\mathbf{U}(i)) = \eta \overbrace{p(\mathbf{Z}(i)|\mathbf{x}(i))}^{\text{sensor model}} \int_{\mathbf{x}(i-1)} \underbrace{p(\mathbf{x}(i)|\mathbf{x}(i-1),\mathbf{u}(i))}_{\text{motion model}} \underbrace{p(\mathbf{x}(i-1)|\mathbf{O}(i),\mathbf{U}(i-1))d\mathbf{x}(i-1)}_{\text{recursive term}} \tag{20}$$

This is realized by a set of particles which predicts the measurements for a given generalized state $\mathbf{x}$, and which updates their probability according to the difference between the real measurements and the prediction. A fixed number of particles with the lowest probabilities have their state regenerated from the particles with the highest probabilities. Hornung in [22] demonstrated how to use this technique with a laser mounted on top of a Nao humanoid robot.

2.5 Data association problem

Problem statement

Definition 2. *Data association problem*
The data association problem *is to decide if the measurement $\mathbf{d}_l^i$ of landmark l in image i and the measurement $\mathbf{d}_m^j$ of landmark m in image j correspond to the same landmark.*
More formally, the data association problem consists in finding a Boolean matrix $\mathbf{D}$ *of size $|\mathcal{L}_i| \times |L_i|$ such that:*

$$\mathbb{1}_{|L_i|} = \mathbf{D}\mathbb{1}_{|\mathcal{L}_i|} \tag{21}$$

where $\mathbb{1}_n$ is a vector of size n and where all elements are equal to one. This problem is very important as it allows to build the graph linking the measurements between each other. A first solution is to associate a landmark l from L_i to a landmark m from $\mathcal{L}_i$ when l is the closest among all the previously seen landmarks.

$$l = argmin_{j \in \mathcal{L}_i} d(\mathbf{d}_j^i, \mathbf{d}_m) \tag{22}$$

Landmarks in key images Key-images are extracted from the flow of images provided by the camera. In a key-image, particular points of the environment called landmarks are detected. A landmark l has a position in space that is noted $X_l \in \mathbf{R}^3$ and a visual appearance. From the image, it is possible to detect a *projection* of the landmark visual appearance according to the robot position with respect to this landmark. Let us denote by L_i the set of landmarks in image i and $\mathcal{L}_i = \{L_0, \ldots, L_{i-1}\}$ the set of all the landmarks detected until considering image i. The problem is then to recover the unknown position $X_l \in \mathbf{R}^3$ of each landmark l from $\mathcal{L}_i$. In the key-images representation $\mathcal{L}_i$ is considered as the map of the environment when the landmarks positions are estimated.

In order to realize this estimation, landmarks are detected in the image by a *feature detector*. A popular feature detector is the Harris detector. It provides a position in the image plane $\mathbf{q}_l^i = (x_l^i, y_l^i)^T$ for the image I_i and landmark l. For each landmark a *feature descriptor* $\mathbf{d}_l^i$ is build by extracting a local patch from image I_i and computing a vector in $\mathbb{R}^n$. It is a compact representation of the landmark visual appearance. Popular feature descriptors are for instance SIFT [29], or SURF [2]. Often they are completed with other simple features such as color information or disparity for stereoscopic system. Feature descriptors are usually associated with a distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ to compare any two of them.

Bag of words Using only appearance to detect landmarks is performed using the previously introduced feature descriptors. A statistical study is performed over the set of measure $\mathbf{d}_k^i$ to compute the probability of occurence of this landmark, also called a *word*. In order to lower the number of possible words, k-means clustering is usually performed over the set of landmarks. It is also possible to compute the mutual information between the landmarks in an image to find out the most discriminative ones. This allows to detect repetitive patterns and to select the landmarks that are easier to track. The matrix relating the appearance-based correlation between landmarks is usually called the confusion matrix. To improve further the robustness of detecting landmarks, based only on image information, it is possible to compute the probability of co-occurence.

When the exploration of an unknown environment is realized with a localization device, it is possible to compute the most probable location of the robot using only appareance [7]. This is especially useful when the robot has started and has to recognize the place. This is the *kidnapped robot problem*.

Appearance and geometry Using an appearance-based criteria alone is not enough when the environment includes repetitive patterns, which is often the case in buildings. Descriptors in addition are not all invariant to scale, rotation and angle of view. As the data association problem has to take into account both geometry and the visual descriptor, there is a large body of literature on the two subjects. There are two instances of the data association problem which can be solved by different formulation. The first one is *visual odometry*, when one wants to evaluate the rigid motion between two consecutive images.

The second problem is the detection of *loop-closure*. A loop-closure occurs when the robot, after exploring an unknown part of the environment, perceives again landmarks in its map. When there are several possible candidates, the system has to evaluate the various solutions and choose the most probable one.

Visual odometry When a collection of pairs of landmarks are detected between two successive images i and $i+1$ it is possible to estimate the rigid motion between the two images that matches at best the measurement. For instance let us consider the position $\mathbf{q}_l^{i+1} = (u_l^{i+1}, v_l^{i+1})^T$ of landmark l in image $i+1$. Assuming that the position of camera at image i is the origin, then the rigid motion noted by a rotation matrix and a translation $\{\mathbf{R}, \mathbf{T}\} \in \mathbb{SE}(3)$, expresses:

Rigid Motion

$$argmin_{\{\mathbf{R},\mathbf{T}\}\in\mathbb{SE}(3)} \sum_{l\in L_{i+1}} ||\mathbf{q}_l^{i+1} - p\left(\mathbf{K}\begin{pmatrix}\mathbf{R} & \mathbf{T}\end{pmatrix}\begin{pmatrix}X_l^x\\X_l^y\\X_l^z\\1\end{pmatrix}\right)|| \tag{23}$$

Naturally the result of Eq. 23 is valid only if $\mathbf{q}_l^{i+1}$ is really the projection of landmarks L_{i+1}. In order to evaluate it, a powerful method to use is the *random sample consensus method* (RANSAC). It consists in randomly choosing a set of n different landmarks measures to compute a hypothetical rigid motion solution of Eq. 23. Then this rigid motion is used to compute the projection of all the other landmarks measured in the image. The difference between the projection and the real measurements gives an indication on the assumption validity. Because it exists analytical solutions when considering a small number of parameters ($n = 8$ [28] $n = 7$[18], $n = 6$[43,44], $n = 5$[51] or even $n = 1$[47] when the robot behaves like a nonholonomic platform), the assumption evaluation can be done efficiently. This technique is very popular for real-time implementation of visual odometry. Indeed, it is possible to continue the procedure until a time limit is reached and the result is the most probable solution [38].

Loop-closure Loop closure is very important to enforce the structure of a map. In Fig. 5 when the camera is seeing again the landmark L_2, this new constraint decreased drastically the error along the camera trajectory. However if the constraint introduced is wrong, for instance by linking the camera position to landmark L_4 because it has a appareance similar to L_2, the consequence on the map reconstruction might be disastrous. A technique to fix this problem is to evaluate the consistency of the map generated by the assumption. If a loop closure detection does not explain properly the measurements, then it is likely to be false compared to a more consistent loop closure.

2.6 Object models

Object can also be used to find the position of a robot. It is even mandatory in order to perform manipulation tasks in interaction with the object. The problem

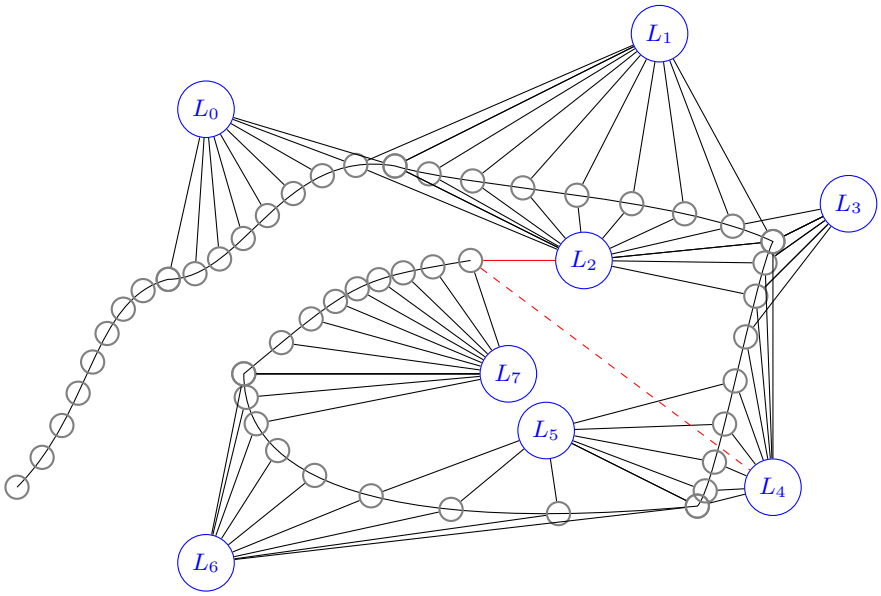


Fig. 5: Camera trajectory and landmarks visibility. The red edges show the occurence of a loop-closure. In both cases, the appareance of the landmark is identical. The plain edge corresponds to the right data association. The dashed one is a wrong one.

is often cast in a similar manner to the *Rigid Motion* problem (Eq. 23) when the features depends on what is the most discriminatory with the considered object. ViSP is a powerful software toolbox which implements all an Expectation Maximization algorithm which solves this problem and uses an estimation of the robot localization to track in real-time the object model. A redudancy of the visual features helps in avoiding singularities which degrade the localization precision. A localization algorithm proposed by Gonzalez et al. [8] is using a particle filter approach to find the 6D localization for the ARMAR robot family. It is using the model of the kitchen in which the robot is evolving. Another example is to link object with affordance to be able to plan manipulation. This was integrated during the DRC by the MIT Team [13].

2.7 Dense representation

Voxel grid In this type map space is represented by cells of equal volume. The problem is then to integrate the measurements and compute the probability that a cell in the range of the sensor is occupied, empty or occluded. This representation is particularly suited for range-images provided either by a RGB-D camera, or a stereoscopic system, typically found on humanoid robots. A laser can be also used but due to its weight and volume it is found mostly on human-size

robots. For a given map $\mathcal{M}$, it allows for instance to localize the robot by finding the most appropriate location which matches the measurement. Because this is costly to store, such maps can be represented using octree. A popular package is Octomap [23]. Voxel grid are very important because they provide possible contact surfaces. They can be used to decide which parts of the environment are traversable. Areas where the geometry has a high frequency are unlikely to provide stable contact surfaces, unless sophisticated control algorithms are used. On the other hand smooth areas are much easier to handle. Therefore when planning motion for a humanoid robot, the set of actions to choose from should be adapted according to the smoothness of the geometry. This approach has been implemented in the planner proposed by Hornung [21] to make the humanoid robot NAO able to handle cluttered environments.

Dense-SLAM This approach aims at building a dense representation of the environment. It is realized through the computation of a surface delimiting the outer boundaries of an object.

Dense-SLAM is of high interest for humanoid because they provide not only the obstacles to avoid, but although possible walking surfaces. Having a reliable segmentation of the surface is thus of primary importance. Surface can be reconstructed using the KinectFusion algorithm [36]. This approach, originally developed for a RGB-D sensor, can also be used with stereoscopic information (see for instance Fig.6). It models 3-D surfaces as zero-valued level sets of functions defined over the workspace volume. These functions are referred to as *Truncated Signed Distance Functions* (TSDFs) and they are incrementally built by integrating the depth measurements the sensor provides, frame after frame. TSDFs are defined in the 3D space and their value is the signed distance to the closest surface. Stereo data are usually noisier than the one from RGB-D sensors, but as the source is a passive sensor, it can be used outdoors in sunlight conditions.

Let us consider a volumetric TSDF model (defined over a 3D grid) and refer to it as F_i at iteration i. The basic steps for integrating one new set of disparity measurements at i, to update F_i and the corresponding surface, are the following:

1. Filter the raw depth measurements generated from the sensor (D_i). For that purpose, a bilateral filtering can be used.
2. From these filtered measurements and the prediction of the estimated surface at the previous step, estimate the transformation between the measured surface and the predicted one using the iterative closest point algorithm (ICP) and update the camera pose.
3. Compute a volumetric grid formed from "local" TSDF values F_{D_i}, to which confidence weights W_{D_i} are associated, and integrate them into the global volumetric grid $\{F_i, W_i\}$.
4. Predict a new surface for the next iteration by using ray-casting over the zero-crossings of the fused global volumetric grid $\{F_i, W_i\}$.

The core of this algorithm is the computation and fusion of volumetric grids (i.e., the third step mentioned above). For a 3D point p, expressed in the global

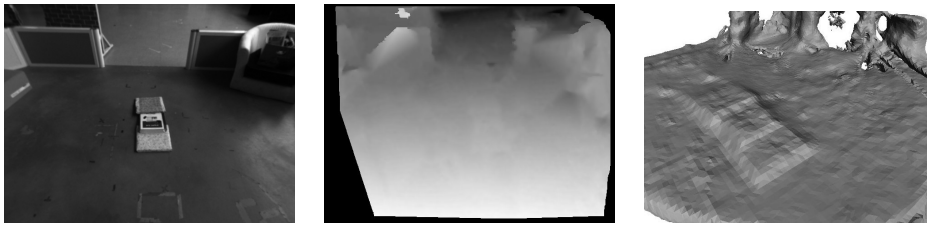


Fig. 6: From a pair of images of the scene in front of the robot (left) a dense disparity map is estimated (middle) and from this disparity map a dense surface integrating the previous frames into the volumetric grid is estimated (right). These images were taken from the HRP-2 stereo vision system with the cameras tilted towards the ground to allow ground reconstruction tasks [26]. It is assumed that a mechanism is implemented to make the robot search for traversable areas [31].

frame g, its value in the current local volumetric grid $\{F_{D_i}, W_{D_i}\}$ is computed as

$$\begin{aligned} F_{D_i}(p) &= \Psi(\lambda^{-1} \left\| t_{g,i} - p \right\| - D_i(x)), \\ W_{D_i}(p) &\propto \cos(\theta)/D_i(x), \end{aligned}$$

with $\lambda = \left\| K^{-1}[x^\top \ \mathbf{1}]^\top \right\|$ and

$$\Psi(\eta) = \begin{cases} \min(1, \frac{\eta}{\mu}) \ \mathrm{sgn}(\eta) & \text{iff } \eta \geq -\mu \\ \text{null} & \text{otherwise} \end{cases}$$

where μ is a truncation distance (a parameter of the algorithm), and $x = \pi([K, \mathbf{1}]T_{g,k}^{-1}p) \in \mathbb{R}^2$ is the image projection of p. K is the 3×3 matrix of intrinsic parameters of the camera, π is the projection operator, $T_{g,k} = \begin{bmatrix} R_{g,k} & t_{g,k} \\ 0 & 1 \end{bmatrix}$ is the pose of the camera at time k in the global frame g, and θ is the angle between the associated pixel ray direction and the surface normal.

The global volumetric grid at i is formed by the weighted average of all individual volumetric grids up to $i - 1$. It can be shown that the optimal grid can be incrementally obtained using a simple pointwise on-line weighted average

$$\begin{aligned} F_i(p) &= \frac{W_{i-1}(p)F_{i-1}(p) + W_{D_i}(p)F_{D_i}(p)}{W_{i-1}(p) + W_{D_i}(p)}, \\ W_i(p) &= W_{i-1}(p) + W_{D_i}(p). \end{aligned}$$

To use this algorithm with stereo data and generate local data D_i, a disparity map from a pair of rectified images is estimated, from which the depth map D_i is derived assuming that the stereo rig is completely calibrated. The literature of algorithms devoted to disparity maps estimation is huge, but when a real-time one is needed for this application, the one proposed in[16] might be used. This algorithm estimates a piecewise disparity map using an initial sparse

disparity map of high textured points as vertices that defines a triangulation of the image. Then, the dense disparity map of each sub-region is estimated using the initial sparse disparity map as a prior in a probabilistic scheme. They are strong connections between the surfarce reconstruction and the voxel grid approach. However they are two different mathematical objects: one providing a probability, the other describing a boundary.

3 Visual navigation

3.1 Planning

The problem solved by planning is to find a sequence of feasible control vector $\mathbf{u}(t)$ such that the robot is going from an initial configuration to a final configuration without colliding with obstacles, while making contacts with appropriate parts of the environment. The problem is quite complex because a humanoid robot needs to find a sequence of contacts to perform the locomotion task while respecting its physical constraints described in paragraph.1.3. In addition when the problem is solved at a certain time t, it is done with a model of the world $\mathbf{v}(t)$ including uncertainities. For this reason, the plan has to include a form of regulation to handle such uncertainities when it is executed.

Therefore planning for humanoid robots is in general using two components: a contact planner, essentially a foot-step planner which provides the contacts to realize to a controller.

Quasi-static pose transition A first approach is to find a sequence of contacts for which exists a collision-free transition of quasi-static poses between the contacts that respects all the constraints of the robot. The transition between the contacts can be realized by linear interpolation between the two poses, and projected on the submanifold by computing the appropriate tasks and constrains. Then the resulting trajectory can be accelerated while keeping the robot balance.

A common other strategy is first to plan footsteps, generate a dynamically stable trajectory for an equivalent point mass model and third to deduce a set of joint trajectories. This trajectory is tested against the environment, if collision are detected then the footsteps are modified.

Both approaches have a high power of expressivness but can be computationally intensive. This can be overcome by using a database of precomputed poses. For an arbitrary pose, the system is then trying to a find a solution starting from the closest one in the database.

Discrete set of actions When real-time is required, a practical approach is to consider that contact transition is decided in a reduced configuration space $\mathcal{C}$ associated with a limited number of actions $\mathcal{A}$. In addition, the actions should be performed on a subset of $\mathcal{C}$ collision free. This subset is called $\mathcal{C}_{free}$. An early work proposed by Okada and al. [40] used this approach. It was also

reused by Gutman and al. [17] on QRIO to consider four kinds of actions: $\mathcal{A} = \{forward, backward, right\ side, left\ side\}$. In [17] the geometry of the robot is simplified to a circle in 2D and to a cylinder in 3D. Therefore the configuration space is defined as $\mathcal{C} = \mathcal{X} \times \mathcal{Y} \times \Phi$ where $\mathcal{X}, \mathcal{Y} = \{i.cs \mid 0 \leq i < n\}, \Phi = \{i.45\deg; \mid 0 \leq i < 8\}$, cs is the cell size. The model of the environment is a 2.5D grid map which is used to decide which areas are traversable $\mathcal{C}_{free}$. It uses the A-$*$ algorithm to compute a feasible path from a starting configuration $c_{start}, c_{goal} \in \mathcal{C}_{free}$ over this map. For this the following cost function is considered:

$$f(c_{start}, \pi) = g(c_{start}, \pi) + h(succ(c_{start}, \pi)) \tag{24}$$

with $\pi = a_1 \cdots a_m$ the m actions currently decided by the A-$*$ algorithm, $succ(c, a_1, \cdots, a_m) = succ(succ(c, a_1), a_2, \cdots, a_m)$ the recurvise action execution, with g defined as follows:

$$\begin{aligned} g(c, a_1) &= g_a(a_1) + g_{t'}(succ(c, a_1)) + g_o(succ(c, a_1)) \\ g(c, a_1, a_2, \cdots, a_m) &= g(c, a_1) + g_c(a_1, a_2) + g(c', a_2, \cdots, a_m) \end{aligned} \tag{25}$$

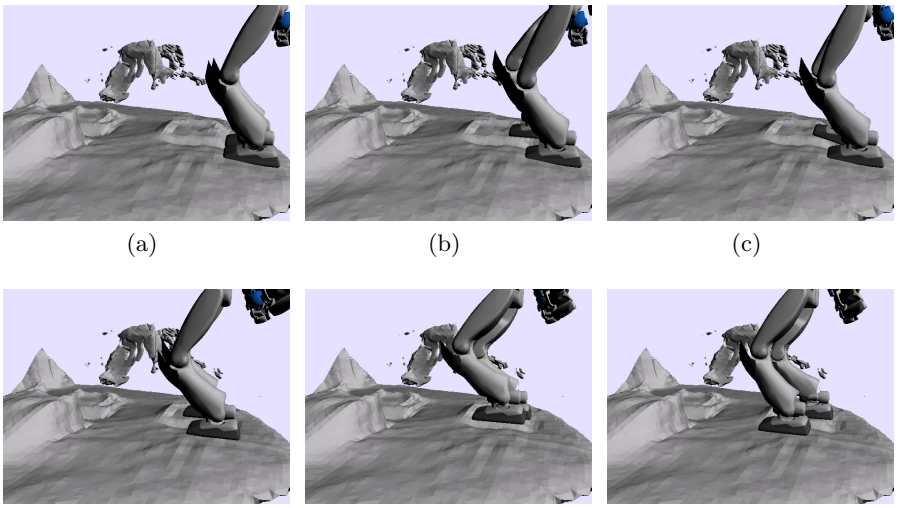
where g_a is cost related to the action (i.e. going forward is less costly than sideways), $g_{t'}$ depends on the map (i.e. walking over an obstacle is more costly than going forward), g_c is a constant for changes in action, g_o is favorizing path away from obstacles. g here is used to evaluate the cost of the current policy π. The function h is a heuristic to evaluate the cost of reaching the goal from the current state to be explored.

Recently Hornung [21] proposed an inflation method which adds a weight on the function h or the heuristic over the future to change the exploratory behavior of the greedy search performed by A-$*$. This kind of approach is complete over the space of configuration and actions. However it does not fully represent the whole set of motions that the robot is able to perform. Therefore if the algorithm does not find a path, it does mean that there is no feasible path for the robot. A refinement of this approach is proposed by Chestnutt in [5] which consists in trying to solve an optimization problem when no solution is found over the set of predefined actions.

Interestingly, the optimization problem solved is usually equivalent to the controllers used on the robots. This is an interesting compromise between the full continuous problem cast in Eq. (1)-(3), and a branch-and-bound approach such as the A* algorithm solving Eq. 24. Ideally if a compact representation of the geometry spanned by the controllers in its local domain of convergece is available, then the plan can be found over this compact representation. When the set of actions is fixed this is indeed the case as depicted in Fig. 3. As obtaining this compact representation is still an open issue, a more common approach is to try to have very efficient optimization problem which solves at the same time the motion control problem and the choice of the contacts.

Mixed integer and continuous formulations have been demonstrated on the Atlas robot [9,27]. Other extensions keeping a quadratic formulation with linear constraints have been proposed to extend the LIPM classical formulation [19]. Fig.7 shows the result of using [19] with the environment reconstructed with a

dense approach depitec in Fig.6. The controller is based on an Operational Space Inverse Dynamics approach [45].



(a) (b) (c)

(d) (e) (f)

Fig. 7: HRP-2 walking on an obstacle.

Several algorithms have been proposed in simulation with impressive results on general cases [35,53]. The open question is now to see if they can be applied to real humanoid robot.

Visual Navigation Apart from the impressive literature from the humanoid or standard platform Robocup league, there is a number of noticeable work on making humanoid robots such as Nao moving using vision in the loop. Classically, navigation is done through a phase of localization, and map building on top of which footstep planning is realized. The footsteps are then given to a motion generation system controlling the robot to perform the footsteps. The planning phase which aims at finding a policy to plan a path between two positions given in 2D, or in 3D, can be for instance replaced directly by the minimization of a visual criteria. It can be for instance a criteria which guarantees that the robot is following a corridor [14]. The criteria can be also to follow a visual path [10]. The Pepper robot for instance uses a topological representation coupled with a compass to perform visual navigation with low computational power [55].

4 Conclusion and perspectives

In this chapter we have briefly outlined the different approaches involved in visual navigation. The presentation decoupled localization and planning as it is currently the most common approach. However it is interesting to note that there is a growing number of works coupling estimation and model predictive control using a unified approach [24,15]. The main difficulty is probably to perform efficient implementation able to cope with the large number of variables of the generalized state. In addition, whereas in large number of applications the main motivation is to reach a specific position, it is very likely that the behavior with higher semantic is of interest to ease the handling of the robot. For instance giving a visual path, or simply asking the robot to follow a corridor or go through a door will simplify considerably the problem of guiding a humanoid robot.

The author work was funded by the European Commission through the EUROC Project.

5 Recommended Readings

[1] A. Andreopoulos, H. Wersing, H. Janssen, S. Hasler, J.K. Tsotsos, and E. Körner. Active 3d object localization using asimo. *IEEE Transactions on Robotics*, 1:47–64, 2011.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. 2006.

[3] C. Bishop. *Pattern recognition and Machine Learning*, chapter Graphical Models. Springer, 2006.

[4] J. Chesnutt, M. Lau, G. Cheung, J. Kuffner, J. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, pages 631–636, April 2005.

[5] J. Chestnutt. Navigation and gait planning. In K. Harada, Eiichi E. Yoshida, and K. Yokoi, editors, *Motion Planning for Humanoid Robots*, pages 1–28. Springer London, 2010.

[6] A. I. Comport, R. Mahony, and F. Spindler. A visual servoing model for generalised cameras: Case study of non-overlapping cameras. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, pages 5683–5688, 2011.

[7] M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *Int. Journ. of Robotics Research*, 27(6):647–665, 2008.

[8] T. Asfour D. Gonzalez-Aguirre, M. Vollert and R. Dillmann. Robust real-time 6d active-visual localization for humanoid robots. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[9] R. Deits and R. Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2014.

[10] J. Delfin, H. M. Becerra, and G. Arechavaleta. Visual path following using a sequence of target images and smooth robot velocities for humanoid navigation. In *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2014.

[11] M. Dissanayake, P. Newman, S. Clark, H.-F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Tran. on Robotics*, 17(3):229–241, Jun 2001.

[12] C. Dune, A. Herdt, O. Stasse, P.-B. Wieber, and K. Yokoi. Canceling the sway motion in visual servoing for dynamic walking. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[13] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C.-P. DArpino, R. Deits, M. DiCicco, D. Fourie, T. Koolen, P. Marion, M. Posa, A. Valenzuela, K.-T. Yu, J. Shah, K. Iagnemma, R. Tedrake, and S. Teller. An architecture for online affordance-based perception and whole-body planning. *Int. Journ. of Field Robotics*, 32(2):229–524, 2015.

[14] A. Faragasso, G. Oriolo, A. Paolillo, and M. Vendittelli. Vision-based corridor navigation for humanoid robots. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, pages 3190–3195, 2013.

[15] M. Garcia, O. Stasse, J.-B. Hayet, C. Dune, C. Esteves, and J.-P. Laumond. Vision-guided motion primitives for humanoid reactive walking: decoupled vs. coupled approaches. *Int. Journal of Robotics Research: Special Issue on Vision*, 2014.

[16] A. Geiger, M. Roser, and R. Urtasun. Efficient large-scale stereo matching. In *Asian Conf. on Computer Vision*, 2010.

[17] J.-S. Gutmann, M. Fukuchi, and M. Fujita. Real-time path planning for humanoid robot navigation. In *Int. Joint Conf. on Artificial Intelligence*, page 2005.

[18] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision.* 2nd ed., Cambridge, UK: Cambridge University Press, 2004.

[19] A. Herdt, D. Holger, P.B. Wieber, D. Dimitrov, K. Mombaur, and D. Moritz. Online walking motion generation with automatic foot step placement. *Advanced Robotics*, 24:719–737, 2010.

[20] A. Herdt, N. Perrin, and P.-B. Wieber. Walking without thinking about it. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 190–195, 2010.

[21] A. Hornung. *Humanoid Robot Navigation in Complex Indoor Environments.* PhD thesis, Albert-Ludwigs-Universitt, Freiburg, Germany, 2014.

[22] A. Hornung, K. M. Wurm, and M. Bennewitz. Humanoid robot localization in complex indoor environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1690–1695, 2010.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robot*, 3:189–206, April 2013.

[24] V. Indelman, L. Carlone, and . Dellaert. Planning under uncertainty in the continuous domain:a generalized belief space approach. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[25] M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Workshop on the Algorithmic Foundations of Robotics*, 2010.

[26] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, and T. Isozumi. Humanoid robot hrp-2. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, volume 2, pages 1083–1090, 2004.

[27] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Under Review*, 2015.

[28] H. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–125, 1981.

[29] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(4):91–110, 2004.

[30] A. De Luca. Visual servoing.

[31] D. Maier, C. Lutz, and M. Bennewitz. Integrated perception, mapping, and footstep planning for humanoid navigation among 3d obstacles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013.

[32] A. Majumdar, A. A. Ahmadi, and R. Tedrake. Control design along trajectories with sums of squares programming. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[33] M. Meilland and A. I. Comport. On unifying key-frame and voxel-based dense visual slam at large scales. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3677–3683, 2013.

[34] H. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2), 1988.

[35] I. Mordatch, E. Todorov, and Z. Popovic. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4), 2012.

[36] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Washington, DC, USA, 2011.

[37] K. Nishiwaki, J. E. Chestnutt, and S. Kagami. Autonomous navigation of a humanoid robot over unknown rough terrain using a laser range sensor. *Int. Journ. of Robotics Research*, 31(11):1251–1262, 2012.

[38] D. Nistér. Preemptive ransac for live structure and motion estimation. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 199–206, 2003.

[39] G. Nuetzi, S. Weiss, D. Scaramuzza, and R. Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of Intelligent and Robotic Systems*, 61:287–299, 2011.

[40] K. Okada, Y. Kino, F. Kanehiro, Y. Kuniyoshi, M. Inaba, and H. Inoue. Rapid development system for humanoid vision-based behaviors with real-virtual common interface. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 2515–2520, 2002.

[41] G. Oriolo, A. Paolillo, L. Rosa, and M. Vendittell. Vision-based trajectory control for humanoid navigation. In *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, pages 118–123, 2013.

[42] N. Perrin, O. Stasse, L. Baudouin, F. Lamiraux, and E. Yoshida. Fast humanoid robot collision-free footstep planning using swept volume approximations. *IEEE Transactions on Robotics*, 28(2):427–439, 2012.

[43] J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *Photogrammetric Record*, 88:589–599, 1996.

[44] O. Pizarro, R. R. Eustice, and H. Singh. Relative pose estimation for instrumented, calibrated imaging platforms. In *Digital Image Computing: Techniques and Applications*, 2003.

[45] O. E. Ramos, G. Mauricio, N. Mansard, O. Stasse, J.-B. Hayet, and P. Soueres. Towards reactive vision-guided walking on rough terrain: An inverse-dynamics based approach. *Int. Journal of Humanoid Robotics*, 11:1441004–1441026, 2014.

[46] F. Saïdi, O. Stasse, and K. Yokoi. *Recent Progress in Robotics Viable Robotic Service to Human*, chapter Active Visual Search by a Humanoid Robot, pages 171–184. Springer-Verlag, 2007.

[47] D. Scaramuzza. Performance evaluation of 1-point-ransac visual odometry. *Journal of Field Robotics*, 28:792–811, 2011.

[48] D. Scaramuzza. *Computer Vision: A Reference Guide*, chapter Omnidirectional Camera. Springer, 2014.

[49] J. F. Seara and G. Schmidt. Intelligent gaze control for vision-guided humanoid walking: methodological aspects. *Robotics and Autonomous Systems*, 48(4):231–248, 2004.

[50] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar. Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, pages 3200–3205, 2008.

[51] H. Stewenius, C. Engels, and D. Nister. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, pages 284–294, 2006.

[52] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[53] P. Wensing and D. E. Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, pages 3103–3109, 2013.

[54] T. Whelan, M. Kaess, H. Johannsson, M.-F. Fallon, J.-J. Leonard, and J.-B. McDonald. Real-time large scale dense rgb-d slam with volumetric fusion. *Int. Journ. of Robotics Research*, 34(4-5):598–626, 2015.

[55] E. Wirbel, S. Bonnabel, A. de La Fortelle, and F. Moutarde. Humanoid robot navigation: Getting localization information from vision. *J. Intelligent Systems*, 23(2):113–132, 2014.