Humanoid Navigation using a Visual Memory with Obstacle Avoidance[⋆]

Josafat Delfin[1], Héctor M. Becerra[2] and Gustavo Arechavaleta[1]

[1]*Robotics and Advanced Manufacturing Group, Centro de Investigación y de Estudios Avanzados del IPN (CINVESTAV), Saltillo, Coah. Mexico. E-mail:*`(josafat.delfin,garechav)@cinvestav.edu.mx`

[2]*Computer Science Department, Centro de Investigación en Matemáticas (CIMAT), C/ Jalisco S/N, Col. Valenciana. C.P. 36023, Guanajuato, Gto., Mexico. E-mail:* `hector.becerra@cimat.mx`

Abstract

We present a complete humanoid navigation scheme based on a topological map known as visual memory (VM), which is composed by a set of key images acquired offline by means of a supervised teaching phase (human-guided). Our autonomous navigation scheme integrates the humanoid localization in the VM, a visual path planner and a path follower with obstacle avoidance. We propose a pure vision-based localization algorithm that takes advantage of the topological structure of the VM to find the key image that best fits the current image in terms of common visual information. In addition, the visual path planner benefits obstacle-free paths. The VM is updated when a new obstacle is detected with an RGB-D camera mounted on the humanoid's head. The visual path following and obstacle avoidance problems are formulated in a unified sensor-based framework in which, a hierarchy of tasks is defined, and the transitions of consecutive and hierarchical tasks are performed smoothly to avoid instability of the humanoid. An extensive experimental evaluation using the NAO platform shows the good performance of the navigation scheme.

Keywords: Visual navigation, humanoid robots, visual memory, obstacle avoidance

1. INTRODUCTION

One of the main objectives to be reached by a humanoid is to mimic human navigation strategies. An important feature of the human brain is the ability to visually memorize key scenes of previously visited places for facilitating a subsequent navigation in the same place [1]. The idea of using a sequence of key images for robot navigation was early introduced in [2], where the visual memory (VM) was called view-sequenced route representation. A VM is a topological map that represents an environment by a set of key images [3, 4], which are typically organized as a directed graph where each node is a key image and the edges provide information of a relation between key images. Once this representation of the environment is known, the problems of robot localization, path planning and autonomous navigation can be solved.

Visual servo control schemes are built upon the sensor-based control approach, taking advantage of the rich information of images acquired by a vision system to command the

[⋆]The first two authors were supported in part by CONACYT [grant 220796].

motion of a robot. However, the scope of visual-servo controllers is limited to positioning tasks where the visual scene is partially observed from the initial and target locations [5, 6, 7]. This is due to the field-of-view constraint of typical vision systems [8]. This local behavior appears in any of the two classical approaches of visual servoing: the position-based scheme (PBVS), where the estimation of some 3D pose parameters is needed; and the image-based scheme (IBVS), where direct feedback of image features is used [9]. The extension suggested in [2, 10] takes advantage of a sequence of target images (VM), which share visual information between each pair of consecutive images, and the image acquired at the initial location shares information with one of the target images. In this manner, the robot enlarges its workspace as the images associated to the initial and final locations do not need to share information. This strategy has been mainly exploited for navigation of wheeled mobile robots [2, 4, 11, 12].

In this paper, we propose a humanoid navigation strategy based on a VM, which encloses robot localization, visual path planning and path following with obstacle avoidance. From a minimum number of detected point features, the proposed strategy computes continuous velocities that can be applied in indoor environments with natural scenes, like corridors, uncluttered or cluttered environments.

This paper is an incremental work with respect to our previous results in [13] and [14]. In the first reference, we have proposed a visual path following scheme for humanoid robots that is generic in the sense that it can be used for different types of visual servo controllers. In the second reference, we have extended the scheme by solving the initial humanoid localization problem in the VM to plan a visual path when the VM has several forks and/or cycles. As *extensions* to those works, in this paper we propose *an enhanced pure vision-based localization algorithm* that first finds a neighborhood of key images around the image currently observed by the robot, to later perform a local search for the key image that best fits the current image in terms of common visual information. In addition, we incorporate *a versatile obstacle avoidance strategy* that is aided by depth information from an RGB-D camera mounted on the robot's head. The strategy deals with static obstacles that appear during the autonomous navigation, and that were not present during the teaching phase to generate the VM. Besides, the unexpected obstacles are localized in the VM, and they are taken into account for subsequent navigation tasks.

The main *contribution* of this work is *a novel vision-based navigation scheme for humanoids formulated in a unifying framework of consecutive and hierarchical tasks (visual servoing and obstacle avoidance tasks), for which, the transitions between tasks are performed using smooth functions to keep the balance of the humanoid platform.* The generation of continuous velocities that are given to the walking pattern generator (WPG) is particularly important for visual control of humanoids [7]. Task transitions occur when a new target image is given, or when an obstacle is detected for stepping over or circumvent it while maintaining the visual target in sight during the navigation. The mobility of the humanoid robot is exploited to walk laterally when it is needed as well as to step over obstacles. Thus, no motion constraints are imposed to the robot. An extensive experimental evaluation using the NAO platform shows the good performance of the navigation scheme.

Our proposal aims to use 2D information as much as possible, such that the only available information for navigation is a set of key images and the current view. The generation of the VM is done offline, i.e., the key images to form the VM are acquired and selected by means of a supervised teaching phase (human-guided). Depth information is just a complement to facilitate the obstacle avoidance task. We have left as future work the detection of obstacles relying only on monocular vision. The main advantage of using 2D information instead of 3D is the reduction of computational requirements, which are poor in commercial and small-sized humanoid platforms like NAO, since there is no need of generating and updating a complex spatial map. For instance, in the proposed localization method no extra information is needed, like odometry or sensor fusion.

The rest of the paper is structured as follows: Section 2 presents a brief account of related work. Section 3 presents an overview of the whole navigation strategy. Section 4 details the structure of the graph that represents the VM. Section 5 describes the proposed localization algorithm and the path planning stage. Section 6 presents the proposed visual path following scheme with smooth transitions of visual tasks. Section 7 describes the obstacle avoidance strategies. Section 8 presents the experimental evaluation of the stages of the method and its integration. Finally, Section 9 gives some concluding remarks and ideas of future work.

2. Related work

The use of VM has made possible to overcome the local behavior of visual servo control schemes. The navigation based on a VM has been mainly used for wheeled mobile robots. For instance, a local Euclidean reconstruction is used in [4] to avoid building a complete map. Geometric reconstruction is used in [11] to predict the feature positions, and it allows recovery of features tracking failures. An image-based scheme reported in [12] makes use of direct feedback from a visual geometric constraint. In [15], the integration of a laser range finder in the visual navigation helps to avoid unexpected obstacles. Different from other robots, humanoids are able to perform several tasks at the same time [16], which represents one of their main advantages.

Vision-based control of humanoid robots is an interesting and challenging problem due to the inherent dynamic balance and the undesired sway motion introduced by bipedal locomotion [5, 6]. Solutions to the visual servoing problem for local positioning of humanoid robots have been reported in [5, 6, 7]. These control schemes exploit the reactive WPG proposed in [17], which provides automatic generation of foot placements from desired values of linear and angular velocities of the robot's center of mass (CoM). In [5] and [7], a visual servoing scheme gives the velocity reference for the CoM as the main input to the WPG. Such schemes are considered decoupled approaches since the visual controller is independent of the WPG. In contrast, a coupled approach has been proposed in [6], where visual constraints are directly introduced in the WPG.

The extension of visual servo controllers to deal with humanoid navigation in indoor environments has important applications in service robotics for surveillance and assistance. In the literature, a camera onboard the robot has been used to achieve vision-based navigation, which requires a larger displacement than the local visual servoing task [10, 18, 19]. A landmark-based navigation approach that integrates motion planning through geometric primitives and visual servoing tasks is described in [20]. Although the motion planner returns obstacle-free paths, the locomotion model is not capable to step over obstacles. A trajectory tracking control in the Cartesian space, based on vision aided by odometry, is proposed in [18]. The estimation of the humanoid's spatial location considers data fusion from different sensors, similarly than in [21].

Pure vision-based humanoid navigation strategies have also been proposed [10, 22, 19]. In particular, [22] suggests the use of landmarks like 2D bar codes while [19] proposes the use of vanishing points. Both schemes consider environments with corridors. The extension of the view-sequenced route representation for humanoid navigation is reported in [10]. The evaluation is also carried out in a corridor with no initial localization and the controller is based on template correlation to decide basic motion actions. Thus, our proposed scheme can be used in less restrictive indoor environments with natural scenes unlike the referred works [10, 22, 19]. Moreover, our method considers the holonomic nature of humanoid robots, in contrast to previous works [20, 18, 19], which constraint the robot motion as a unicycle model.

An important issue for humanoid navigation is the ability of avoiding obstacles. In structured indoor scenarios, the scheme reported in [23] uses line-based scene analysis and feature tracking to step over or walk around obstacles. Vision aided by range sensors has also been used for humanoid navigation with obstacle avoidance in [24]. In that work, obstacles are identified by means of laser data and that information is then used to train visual classifiers that are later applied to the images during autonomous navigation. Some works focus on the generation of stable dynamic movements in scenarios with obstacles by stepping over them [25, 26]. These works were conceived for human-sized humanoid platforms, like the HRP-2 robot, and a priori knowledge of obstacles dimensions. Recently, an extended version of the WPG of [17] has been introduced in [27], which considers the feet orientation, and it is applied for obstacle avoidance using the HRP-2 robot. However, the underlying nonlinear model predictive control must solve a nonlinear program with a time horizon, which is computationally expensive. Our navigation strategy proposes a simpler method for obstacle avoidance than the ones used in previous works, focusing in an adequate management of hierarchical tasks. Monocular vision aided by data from encoders has been suggested in [28]. The scheme uses optical flow to make a representation of maze-like environments with obstacles to identify the free space. In [29], the humanoid navigation in a cluttered environment has been addressed, considering an application where the robot transports a load. The navigation scheme relies on a 3D representation of the environment obtained using SLAM, from which, collision-free trajectories are computed.

To the authors knowledge, the humanoid navigation based on VM has not been previously extended to deal with obstacle avoidance in indoor environments regardless the characteristics of the work space, i.e., uncluttered, cluttered or corridor-like indoor environments.

3. OUTLINE OF THE NAVIGATION STRATEGY

The autonomous navigation framework presented in this work can be divided in four stages: 1) visual memory building, 2) robot localization and path planning, 3) path following and 4) obstacle localization and graph updating. Fig. 1 presents an outline of the complete navigation strategy. In the context of this work, a VM is a directed graph $\mathbf{G} = \{\mathbf{I}, \mathcal{E}\}$, where each node encodes an image of a set $\mathbf{I} = \{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_m\}$ of m key images and $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_p\}$ is the set of p edges where each of them has an associated weight that links a pair of nodes. Recall that we assume that the VM generation is done offline by means of a supervised teaching phase (human-guided). The automatic selection of key images for building such VM represents a problem by itself, and it is not addressed in this work. However, we point out that some methods for keyframes selection have been proposed in the

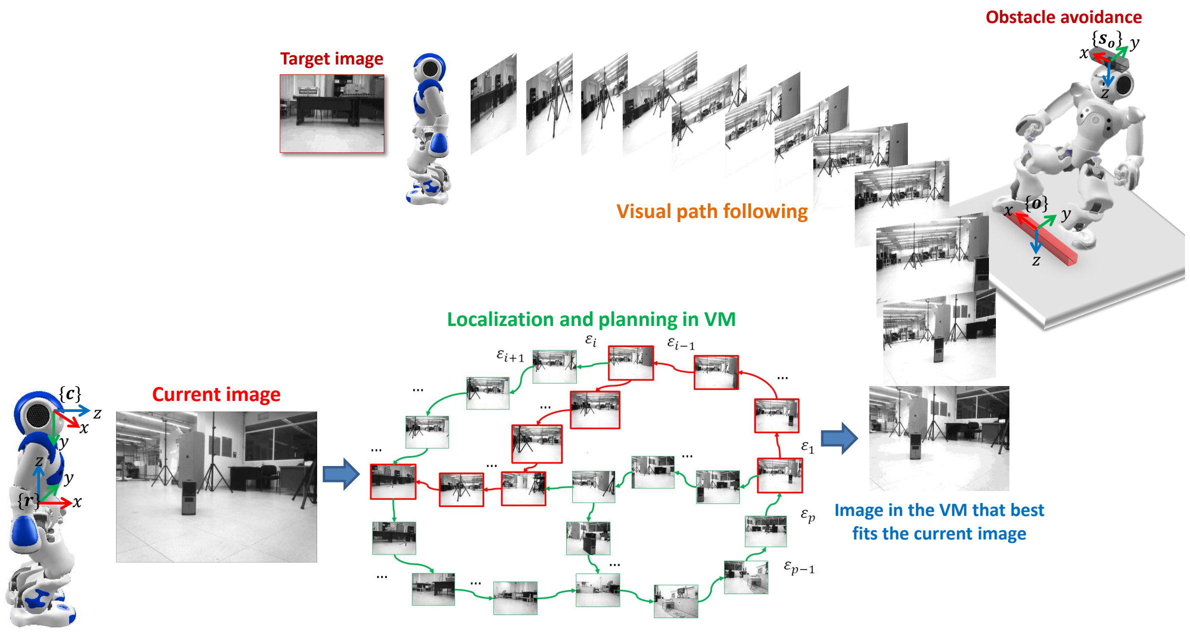

Figure 1: **Outline of the navigation strategy.** Once the VM is available, the robot must localize itself in the VM by finding the image that best fits the current image. Then the visual path to the target image is found by a path planner. The robot navigates autonomously by following the visual path while avoiding static obstacles detected with a range sensor, which were absent during the teaching phase. The notation $\{\cdot\}$ stands for the reference frame of the robot r, camera c, obstacle o and range sensor s_o.

literature related to the field of visual SLAM [30] or video content analysis [31]. The aim of these methods is to ensure a minimum of common visual information between selected images. In this work, we assume that the VM is given as input of the navigation strategy, provided that the keyframes in the VM have enough overlapping of visual features.

The robot localization, which is the second stage of the strategy, consists of finding the key image denoted by $\mathcal{I}_1^*$ in the VM that best fits the current image $\mathcal{I}$ in terms of common visual information. Once $\mathcal{I}$ and $\mathcal{I}_1^*$ are linked, then a key image that belongs to the VM is given as the target $\mathcal{I}_n^*$. The visual path planning is carried out to find the shortest visual path $\mathbf{I}^*$ in $\mathbf{G}$ from $\mathcal{I}_1^*$ to $\mathcal{I}_n^*$ in terms of weights of the edges.

In the path following phase, the robot performs an autonomous navigation to follow the resulting visual path. A visual control scheme guides the humanoid along the sequence of key images until the robot achieves the desired location. The last step of the navigation strategy is the VM updating, which consists of modifying the weights attached to the edges of $\mathbf{G}$ whenever the robot encounters new static obstacles during the visual path following. In particular, if an obstacle is found while the robot is following the piece of visual path from $\mathcal{I}_i$ to $\mathcal{I}_{i+1}$, the weight at the corresponding edge $\mathcal{E}_i$ is increased. Thus, the number N_o of detected obstacles and their location in the VM are memorized. This allows the robot to avoid costly paths with obstacles in a future navigation.

The main steps of the proposed humanoid navigation framework are presented in Algorithm 1, where each function is described in detail in the subsequent sections.

Algorithm 1: visualNavigation allows the autonomous navigation of the robot.

```
   Input: Graph of key images $\mathbf{G} = \{\mathbf{I}, \mathcal{E}\}$, target image $\mathcal{I}_n^*$
   Output: Autonomous visual navigation
1  $\mathcal{I}$ = captureCurrentImage
2  $\mathcal{I}_1^*$ = localizeRobot( $\mathcal{I}$, $\mathbf{I}$ )
3  $\mathbf{I}^*$ = getShortestPath( $\mathcal{I}_1^*, \mathcal{I}_n^*, \mathbf{G}$ )
4  (NAVIGATION, $N_o$, $\mathcal{E}_i$) = pathFollower( $\mathcal{I}$, $\mathbf{I}^*$, $\mathbf{G}$)
5  if $N_o > 0$ then
6      $\mathbf{G}$ = graphUpdating( $N_o, \mathcal{E}_i, \mathbf{G}$ )
7  return
```

4. STRUCTURE OF THE VISUAL MEMORY

As aforementioned, a VM is a graph $\mathbf{G} = \{\mathbf{I}, \mathcal{E}\}$ where the nodes $\mathbf{I}$ encode key images obtained during a teaching phase. It is worth noting that the graph is not always linear, the VM could have several branches and loops. In this section, we complement the description of the graph's structure by defining the weights of the edges between nodes.

The weight attached to an edge represents the cost of traveling from the current node to one of the adjacent nodes, and it is denoted by $\mathcal{E}_i$. We define the weight of an edge in terms of the number of matched interest points and the amount of rotation between neighboring key images as follows:

$$\mathcal{E}_i = \alpha(1 - \overline{s}_i^*) + \beta\overline{\theta u}_i^*, \tag{1}$$

where $\overline{s}_i^*$ and $\overline{\theta u}_i^*$ are the normalized number of matches and the normalized rotation with respect to the vertical axis (perpendicular to the motion plane) between neighboring key images respectively, α and β are weights to favor one of the terms if desired. The cost related to the number of matches $(1 - \overline{s}_i^*)$ means that, the more matches there are between nodes, the lower the cost will be. The cost related to the rotation means that, the more rotation there are between nodes the higher the cost will be.

In previous works [3, 4], the weights of the edges are unitary, i.e., the cost of a visual path is given by counting the number of key images in the route, and an edge exists if a minimum number of point correspondences is achieved to relate neighboring nodes. In this work, the term related to rotation $\overline{\theta u}_i^*$ is included to deal with environments where the richness of point features $\overline{s}_i^*$ is low but sufficient, and the rotation can become more important to determine the cost of a route.

Thus, we estimate the relative rotation θu_i^* between neighboring key images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$, which have associated camera reference frames $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$, respectively. An option to recover the whole relative pose (with translation up to scale) between the frames $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ is by means of the decomposition of a geometric constraint, for instance, the homography matrix [32]. To estimate the homography matrix $\mathbf{H}_i^*$, only the key images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ are needed. The relative transformation between $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ is encoded in the Euclidean homography as [33]:

$$\mathbf{H}_i^* = \mathbf{R}_i^* + \frac{\mathbf{t}_i^*}{d_i^*}\mathbf{n}_i^*, \tag{2}$$

where $\mathbf{R}_i^*$ and $\mathbf{t}_i^*$ are the rotation matrix and translation vector expressed in $\mathcal{C}_{i+1}$, d_i^* is the distance from $\mathcal{C}_{i+1}$ to a plane π, and $\mathbf{n}_i^*$ is the unitary vector expressed in $\mathcal{C}_{i+1}$ normal to

π. According to (2), it is possible to decompose $\mathbf{H}_i^*$ to obtain $\mathbf{t}_i^*$ up to scale and $\mathbf{R}_i^*$. The rotation matrix $\mathbf{R}_i^*$ can then be parametrized by the axis/angle $\theta\mathbf{u}_i^*$. The second element of the vector $\theta\mathbf{u}_i^*$, related to the rotation around the $y-$axis of the camera (perpendicular to the motion plane), provides the required rotation between neighboring key images. In addition, such angular component is used to compute the angular velocity, which corresponds to one of the inputs of the WPG. The other two inputs are longitudinal and lateral velocities as it will be explained in Section 6.

In this work, we rely on the homography model to assign the weights of the edges in the graph, to localize the robot and to formulate the visual control scheme. The homography, as a geometric constraint between images, allows the process of points matching to be robust. This projective model is actually general, since it can be computed for three-dimensional non-planar scenes [32]. Besides, the homography model does not have the issue of being bad conditioned with short baseline, as other geometric constraints like the epipolar geometry [33]. In the scheme of [32], there is no need of verifying the existence of a dominant plane in the scene but a virtual plane is defined by selecting three non-collinear point features in the image. A practical suggestion is to select automatically the three points that maximize the area of the corresponding triangle in both images. Differently from the common homography matrix estimation, the scheme based on a virtual plane needs at least eight points (three reference points and five supplementary) to estimate the model.

Although the homography model is a good option for extracting the required information directly from images in all stages of the proposed approach, the navigation scheme could handle other geometric constraints such as the fundamental matrix. However, the epipolar geometry is ill-conditioned with short baseline and planar scenes, but a model selection between the homography and fundamental matrices could be formulated as in [34].

5. VISUAL LOCALIZATION AND PLANNING

Once the VM is available with the structure described in the previous section, and before starting the autonomous navigation, the robot must localize itself by comparing its current view with the set of memorized key images. Next, given a target image, a path planning stage is needed to find the sequence of key images connecting the image resulting from the localization and the target image. In both stages, we take advantage of the homography estimation for planar and non-planar scenes [32] based on the formulation of a virtual plane.

We propose Algorithm 2 to perform the required appearance-based robot localization. To avoid an exhaustive comparison between the current image $\mathcal{I}$ and each key image in the VM, we propose a strategy to find a reduced neighborhood of key images surrounding $\mathcal{I}$. This process is performed in line 1, function **findNodesNeighborhood**. The underlying algorithmic strategy uses the divide-and-conquer paradigm [35]. First, the current image is compared with each of the "Intersection" nodes, i.e. the nodes where the branches of the graph intersect. Then, each branch of the graph is divided in two to identify the node at the middle of the branch which is compared with $\mathcal{I}$, and in the next iteration each half of the branch is divided again. Thus, the worst case happens when the algorithm iterates until the current image has been eventually compared to all existing nodes in the graph. The search stops when at least one of the comparisons get a number of matches greater than μ. This minimum number of matches ($\mu = 8$) guarantees the computation of the homography [32]. Once the search terminates, a neighborhood of $\widehat{m}$ nodes surrounding this node is selected,

Algorithm 2: localizeRobot finds the most similar key image $\mathcal{I}_1^*$ in the graph with respect to the current image $\mathcal{I}$.

```
Input: Graph of key images $\mathbf{G} = \{\mathbf{I}, \mathcal{E}\}$, current image $\mathcal{I}$, target image $\mathcal{I}_n^*$
Output: Most similar key image $\mathcal{I}_1^*$
1  $\widehat{\mathbf{I}}$ = findNodesNeighborhood( $\mathcal{I}$ , $\mathbf{G}$ )  // SIZE($\widehat{\mathbf{I}}$) = $\widehat{m}$
2  for $j \leftarrow 1$ to $\widehat{m}$ do
3      $matches$ = match( $\mathcal{I}$, $\widehat{\mathbf{I}}[j]$ )
4      if $matches > \mu$ then
5          $\mathbf{H}_j$ = computeHomography( $matches$ )
6          $(\mathbf{R}_j, \mathbf{t}_j)$ = decomposeHomography( $\mathbf{H}_j$ )
7          $\mathbf{t}_{\pm j}$ = computeDirection( $\mathbf{t}_j$ )
8          $\| \mathbf{t}_j \|$ = computeDistance( $\mathbf{t}_j$ )
9          $\mathbf{D}[j]$ = saveImageData( $\mathbf{t}_{\pm j}$, $\| \mathbf{t}_j \|$ )
10 $\mathbf{I}_+$ = selectForwardImages( $\mathbf{D}$ )
11 $\mathbf{I}_-$ = selectBackwardImages( $\mathbf{D}$ )
12 if SIZE($\mathbf{I}_+$) $> 0$ then
13     $\mathcal{I}_1^*$ = selectMostSimilar($\mathbf{I}_+$ )
14 else
15     $\mathcal{I}_1^*$ = selectMostSimilar($\mathbf{I}_-$ )
16 return $\mathcal{I}_1^*$
17 Function selectMostSimilar( $\mathbf{I}_\mu$ )
18     $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ = selectTwoCandidates( $\mathbf{I}_\mu$ )
19     if $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ belong to same branch then
20         $matches$ = match( $\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*, \mathcal{I}$ )
21         $\{\mathbf{H}_1, \mathbf{H}_2\}$ = parametersFixedPlane( $matches$ )
22         $\{\| \mathbf{t}_1 \|, \| \mathbf{t}_2 \|\}$ = computeDistance( $\{\mathbf{H}_1, \mathbf{H}_2\}$ )
23         $\mathcal{I}_1^*$ = selectTheClosest( $\{\| \mathbf{t}_1 \|, \| \mathbf{t}_2 \|\}$ )
24     else
25         $\{\mathbf{I}_1, \mathbf{I}_2\}$ = getShortestPath( $\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*, \mathcal{I}^*, \mathbf{G}$ )
26         $\{\mathbf{l}_1, \mathbf{l}_2\}$ = computePathLength( $\{\mathbf{I}_1, \mathbf{I}_2\}$ )
27         $\mathcal{I}_1^*$ = getImageWithShortestPath( $\{\mathbf{l}_1, \mathbf{l}_2\}$ )
28     return $\mathcal{I}_1^*$
```

and they are collected in a vector $\widehat{\mathbf{I}}$ of selected key images. The next step consists of finding the most similar key image to $\mathcal{I}$ among the reduced set $\widehat{\mathbf{I}}$. For this, **match** finds the matched features between the current image and all key images in $\widehat{\mathbf{I}}$.

The resulting matches allow the computation of the homography matrix and its decomposition in lines 5-6. Consequently, the estimation of the rotation and translation up to scale is obtained. Then, we classify the key images in two groups according to the resultant direction of the estimated translation vector $\mathbf{t}$ that is expressed in the current camera reference frame, which can be forward $\mathbf{I}_+$ or backward $\mathbf{I}_-$ with respect to the current image location. From the third component of $\mathbf{t}$ denoted by t_z, which is pointing towards the motion direction (see $\{c\}$ reference frame in Fig. 1), we assign a value in line 7 as $t_\pm = +1$ if $t_z > 0$ or $t_\pm = -1$ if $t_z < 0$.

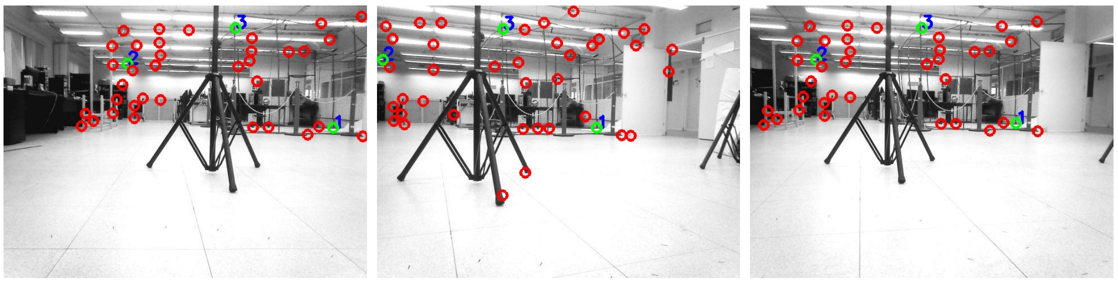

Figure 2: **Fixed virtual plane.** The green circles (marked with number) are selected and maintained as features to form the same virtual plane. **Left.** The first candidate image $\mathcal{I}^*_{\pm 1}$. **Center.** The second candidate image $\mathcal{I}^*_{\pm 2}$. **Right.** The current image $\mathcal{I}$.

Although the vector $\mathbf{t}$ is scaled, its norm $\parallel \mathbf{t} \parallel$ in line 8 gives a notion of distance between key images if the homography computation considers that the distance d_i^* to the plane is constant (see equation (2)). The direction and distance parameters are saved in a vector $\mathbf{D}$ in line 9, from which the forward $\mathbf{I}_+$ and backward $\mathbf{I}_-$ groups of images are obtained (lines 10-11). Using the set $\mathbf{I}_+$, the function **selectMostSimilar** finds (if exists) the most similar key image to the current one in terms of matched points. If there are no forward images, the algorithm selects the closest image among the backward set $\mathbf{I}_-$. Clearly, we prefer to localize the robot even if the most similar key image is behind the current location.

The function **selectMostSimilar** (lines 17-28) selects two candidate images $(\mathcal{I}^*_{\pm 1}, \mathcal{I}^*_{\pm 2})$, i.e., the two images with the highest number of point matches. Then, the algorithm verifies if both candidates belong to the same branch of the graph $\mathbf{G}$. If it is the case, the algorithm selects the most similar key image using the relative distance $\parallel \mathbf{t} \parallel$ from a new homography decomposition. This process computes again $\mathbf{H}$ and $\mathbf{t}$ for the candidate images with respect to the current image, but now the virtual plane used for the homography estimation is fixed (lines 21-22). Thus, the estimated distances are consistent. This process is illustrated in Fig. 2. It consists in matching point features between the three images $\mathcal{I}$, $\mathcal{I}^*_{\pm 1}$ and $\mathcal{I}^*_{\pm 2}$. From the matched features, we select the three points which maximize the surface of the corresponding triangle in the three images [32]. Once the new distances $\{\parallel \mathbf{t}_1 \parallel, \parallel \mathbf{t}_2 \parallel\}$ are obtained, the closest candidate image $\mathcal{I}^*_1$ to the current one is selected (line 23), i.e., the image with smaller $\parallel \mathbf{t} \parallel$.

If the candidate images do not belong to the same branch of the graph, the algorithm selects the most similar key image aided by a path planner that finds the shortest path from the two candidates $(\mathcal{I}^*_{\pm 1}, \mathcal{I}^*_{\pm 2})$ to the target image $\mathcal{I}^*_n$ (lines 25-27). This consideration is needed to discard a localization that might derive in a long path in the navigation stage. The paths associated to the candidate images are denoted $\mathbf{I}_1, \mathbf{I}_2$ and their corresponding lengths are $\mathbf{l}_1, \mathbf{l}_2$. The solution $\mathcal{I}^*_1$ of the localization is the candidate image associated to the path with the minimum length, which is obtained by a function **getShortestPath**. The length of a path is the sum of the values of its edges as defined in (1).

Once the robot is localized, i.e., $\mathcal{I}^*_1$ and $\mathcal{I}^*_n$ are known, the next stage is the visual path planning, which consists of finding the set of key images $\mathbf{I}^* = \{\mathcal{I}^*_1, \mathcal{I}^*_2, \ldots, \mathcal{I}^*_{n-1}, \mathcal{I}^*_n\}$ that links the starting and the desired key images of the VM. Thus, $\mathbf{I}^*$ is the path of minimum length in $\mathbf{G}$ that connects the nodes $\mathcal{I}^*_1$ and $\mathcal{I}^*_n$ and it is obtained by a function **getShortestPath**.

The localization problem addressed in this work can be seen as an image retrieval task.

Several methods have been suggested in the literature for this purpose. For instance, methods based on bag of visual words [36], [37], [38], inverted multi-indexing [39], and nearest neighbor algorithms [40]. These sophisticated techniques can be useful to reduce the computational complexity of the appearance-based localization for very large visual memories, i.e., when thousands of images have to be analyzed. For datasets of around one hundred of images (as the examples in this paper), the proposed method represents a simple ad-hoc approach with very good performance in suitable computational time. Notice that the localization stage is carried out offline, and it is not necessary to have the results very fast.

We consider that all image retrieval methods are prone to provide false-positive detections, of course, also our proposed method, although it was not the case according to the realized experimental evaluation reported in Section 8. We think that a way to reduce the occurrence of false-positive detections could be the use of a local voting scheme considering only a set of neighbors of a candidate image, since more information is better to improve the response with the cost of increasing computation. Therefore, we kept the method as simple as possible but enough to achieve good performance in the localization task.

6. VISUAL PATH FOLLOWING

A visual path is composed by successive key images from the VM, i.e. $\mathbf{I}^* = \{\mathcal{I}_1^*, ..., \mathcal{I}_k^*, ..., \mathcal{I}_n^*\}$ where k denotes the k^{th} key image in $\mathbf{I}^*$. Thus, the path following problem consists of generating the CoM reference velocity to the WPG that regulates the error between the current and the corresponding key images along the visual path. The input of the problem is then the sequence of key images extracted from the VM, and the output is the humanoid walking to track the visual path.

6.1. The sequence of visual tasks

Let us define a visual error as:

$$\mathbf{e}_k = \mathbf{s} - \mathbf{s}_k^* \in \mathbb{R}^6,$$

where $\mathbf{s}$ and $\mathbf{s}_k^*$ are the current and desired visual features, respectively. For a given k, the regulation problem can be treated as a visual servoing task. Here, we propose to solve each visual task by means of a position-based visual servoing (PBVS), however, the problem can be also solved using an image-based visual servo (IBVS) controller as demonstrated in [13]. The choice of a PBVS is due to maintain the same approach in the whole scheme, as the localization stage is based on the estimation of translation and rotation.

The visual features for the PBVS are expressed as:

$$\mathbf{s} = (\mathbf{t}, \theta\mathbf{u}) \in \mathbb{R}^6, \tag{3}$$

where $\mathbf{t}$ is the translation up to scale between the reference frame $\mathcal{C}$ associated to the current camera pose and the k^{th} key image frame $\mathcal{C}_k^*$. The rotation between those reference frames is represented by $\theta\mathbf{u}$ where the axis/angle parametrization is employed. We consider that the translation and rotation are expressed with respect to $\mathcal{C}_k^*$. Therefore $\mathbf{s}_k^* = 0$ and $\mathbf{e}_k = \mathbf{s}$, which is obtained by homography decomposition. We impose an exponential behavior of the error dynamics as:

$$\dot{\mathbf{e}}_k = -\lambda\mathbf{e}_k \text{ with } \lambda > 0. \tag{4}$$

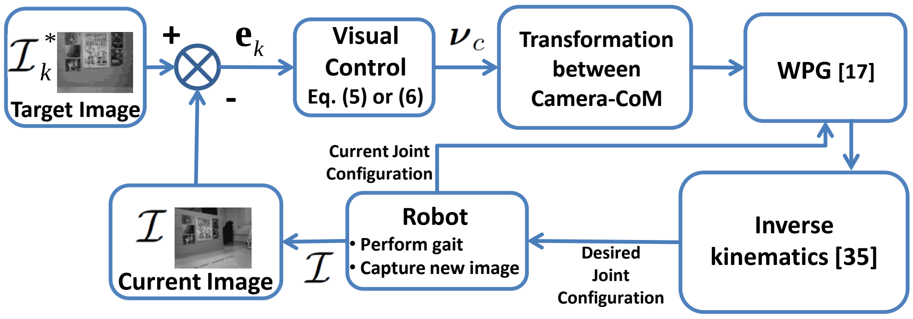

Figure 3: Visual control scheme for humanoids.

Since the translational and rotational velocities are decoupled, the camera twist is computed as in [9]:

$$\boldsymbol{\nu}_c = \begin{bmatrix} \boldsymbol{v}_c \\ \boldsymbol{\omega}_c \end{bmatrix} = \begin{bmatrix} -\lambda\mathbf{t} \\ -\lambda\theta\mathbf{u} \end{bmatrix} \in \mathbb{R}^6. \tag{5}$$

If the camera pose is accurately estimated, the control law (5) yields to an exponentially stable error dynamics (4).

The resulting camera twist (5) could be used as the input to the WPG according to Fig. 3, with the aim of driving the humanoid robot to the location associated to $\mathcal{I}_k^*$ in $\mathbf{I}^*$. However, the abrupt change from k to $k+1$ key images causes discontinuous velocity signals. In addition, it is assumed that the humanoid robot walks on a flat surface which implies that three components of $\boldsymbol{\nu}_c$ are useless. Both issues are treated in the next sections.

6.2. Smooth transitions between key images

The abrupt increment in the error function in (3), when the next key image is given as new target, generates discontinuous camera twists [11, 12]. Since the camera velocity signals are the input reference to guide the humanoid locomotion, it is important to cope with such an undesired behavior. To deal with that, we propose a controller that achieves smooth transitions when the next visual servoing task becomes active. The main ingredient is a time dependent function $h(t)$ that varies from h_0 to 1 within a predefined transition interval. These functions penalize large errors at the beginning of each visual servoing task. The proposed control law written in terms of the PBVS becomes:

$$\boldsymbol{\nu}_c = \begin{bmatrix} -\lambda h\left(t\right)\mathbf{t} \\ -\lambda h\left(t\right)\theta\mathbf{u} \end{bmatrix} \in \mathbb{R}^6. \tag{6}$$

An adequate profile for the transition function $h(t)$ can be obtained by means of the following computation:

$$h(t) = \begin{cases} h_0 + \frac{1}{2}\left(1-h_0\right)\left(1-\cos\left(\frac{\pi(t-t_0)}{t_f-t_0}\right)\right), & t_0 \leq t \leq t_f, \\ 1, & t > t_f, \end{cases}$$

where t_0 and t_f are the initial and final times of the transition function, respectively, and h_0 is a minimum value from which $h(t)$ increases smoothly up to the unity. Therefore, after t_f, the maximum value of the control gain λ is applied. The minimum value of $h(t)$ allows the robot to achieve a continuous motion, i.e., the robot does not stop at each target while the discontinuities in the velocities are significantly reduced. The duration of the transition function $t_f - t_0$ has to be defined adequately to ensure that the maximum control gain is applied at least during some time at each visual task.

The underlying computation of (3) involves the point features $\mathbf{p}_j$ of the current image $\mathcal{I}$ that are matched with the points $\mathbf{p}_j^*$ of the corresponding key image $\mathcal{I}_k^*$ while the humanoid is walking. The matched points are used to compute the control law (6) from the homography decomposition. The switching to the next key image $k+1$ occurs when the mean squared error between the corresponding point features ε remains below a threshold T_ε over a finite number of iterations:

$$\varepsilon_k = \frac{1}{l}\sum_{j=1}^{l} \left\| \mathbf{p}_j - \mathbf{p}_j^* \right\| < T_\varepsilon, \tag{7}$$

where l is the number of corresponding point features for the k^{th} visual task. The same steps are repeated for successive key images in $\mathbf{I}^*$ until the final target image $\mathcal{I}_n^*$ is reached.

Finally, to generate the locomotion from the visual control, we use a WPG that considers automatic footstep placements [17]. In particular, the WPG solves a linear model predictive control problem, and its input is the CoM reference velocity $\dot{\boldsymbol{x}}_r$. The output is used to generate the motion coordination and the gait by means of an inverse kinematics method [41]. To obtain the reference velocity $\dot{\boldsymbol{x}}_r$, the camera twist is expressed in the robot's CoM reference frame as described in [14].

7. OBSTACLE AVOIDANCE

The proposed strategy for avoiding unexpected static obstacles is based on the same task function approach that we have applied to define the sequence of visual tasks. Most of the time, the visual task is the only active task to guide the humanoid walking unless an obstacle is detected. If this occurs, an obstacle avoidance task is smoothly activated as the primary task while the hierarchy of the visual task decreases to become secondary by projecting it onto the null-space of the primary task Jacobian. Depending on the size (width) of the obstacle measured by means of a range sensor mounted on the humanoid's head, the controller decides which obstacle avoidance task should be executed. The coarse approximation of the width and height of detected obstacles are the criteria for the humanoid to circumvent or to step over the obstacle. Actually, we assume that all the paths containing an obstacle are feasible while obstacles are avoided by using one of the two options. The peripheral avoidance deforms the trajectory of the CoM without affecting the visual task. On the other hand, if the robot decides to step over the obstacle then the controller smoothly drives the CoM velocity to zero to reconfigure the humanoid's posture for stepping over the obstacle. To perform such complex motion, several motion tasks become active. The set of active motion tasks are solved by the same hierarchical inverse kinematics method employed to generate the motion coordination for walking. It is important to note that during the obstacle avoidance the visual reference must remain in sight. Once the obstacle avoidance is accomplished, the visual task is smoothly switched to be the primary task, and consequently it generates the necessary CoM reference velocity to reactivate the WPG.

The summary of computations required to guide the humanoid robot through the visual path while avoiding obstacles is given in Algorithm 3. Note that the output of this algorithm corresponds to the number of detected obstacles N_o and the edges $\mathcal{E}_o$ where such obstacles

Algorithm 3: pathFollower allows the humanoid robot to autonomously follow a visual path.

```
   Input: Visual path $\mathbf{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \dots, \mathcal{I}_{n-1}^*, \mathcal{I}_n^*\}$, current image $\mathcal{I}$
   Output: Humanoid walking along the visual path
1  $k \leftarrow 1$      // index of key images in the visual path
2  $N_o \leftarrow 0$    // count the number of obstacles
3  while $k \leq n$ do
4      $\mathcal{I}^* \leftarrow \mathbf{I}^*[k]$
5      $matches$ = match( $\mathcal{I}^*$, $\mathcal{I}$ )
6      $\mathbf{H}$ = computeHomography( $matches$ )
7      $\mathbf{e}_k$ = visualTaskComputation( $\mathbf{H}$ )
8      if an obstacle is detected and $e_o \leq 0$ then
9          if the obstacle is new then
10             $N_o++$
11             $\mathcal{E}_o \cup$ getEdgeFromVM($\mathbf{G}$, $k$)
12         $L_o$ = getObstacleSize()
13         if $L_o > A$ then
14             inactivateWPG() // $h(t)$ varies from 1 to 0
15         else
16             activateAvoidance() // $h_t(t)$ varies from 0 to 1
17     else
18         activateWPG() // $h(t)$ varies from 0 to 1
19         inactivateAvoidance() // $h_t(t)$ varies from 1 to 0
20     $\boldsymbol{\nu}_c$ = computeCameraTwist( $\mathbf{e}_k$ )  // Eq. (15)
21     if $h(t) == 0$ and $e_o \leq 0$ then
22         stepOverObstacle($x_o$, $\mathbf{e}_k$, MotionTasks)
23     robotGait = WPG( $\boldsymbol{\nu}_c$ )
24     $\varepsilon_k$ = meanSquaredErrorComputation( $\mathcal{I}^*$, $\mathcal{I}$ )
25     $\mathcal{I}$ = captureNewImage
26     if $\varepsilon_k < T_\varepsilon$ then
27         $k++$
28 stopHumanoid()
29 return $N_o, \mathcal{E}_o$
```

were found. The obstacle width L_o measured by the humanoid's range sensor is compared to a given value A for deciding which obstacle avoidance behavior should be executed. Thin obstacles are surrounded while obstacles wider than A are passed over. In the first case, the distance from the robot to the obstacle d_o and the difference e_o of d_o with respect to a security distance are used in the algorithm. More details of the variables and functions of the algorithm are explained in the next subsections.

If an obstacle completely obstructs the path, an additional strategy would be needed. For instance, the humanoid could use its visual memory (VM) for replanning a new path from reachable neighbor nodes with respect to its current location. Thus, the connectivity of the graph representing the VM and its number of branches are very important. In this work, however, we have assumed assume that obstacles can be avoided.

7.1. Obstacle localization and graph updating

The localization of an unexpected static obstacle in the VM is important for subsequent navigation tasks. The strategy consists of updating the cost to go from $\mathcal{I}_{i-1}$ to $\mathcal{I}_i$ where the obstacle is detected. Therefore, the cost encoded in the edge $\mathcal{E}_i$ that links both images increases as:

$$\mathcal{E}_i = W\mathcal{E}_i, \tag{8}$$

where W is a high value of weight assigned to the edge. In the case that the humanoid does not detect an obstacle that was previously labeled at $\mathcal{E}_i$, the initial cost is recovered. Clearly, the new cost affects the paths that contain $\mathcal{E}_i$. By means of the graph updating mechanism, next queries in the VM will return less costly visual paths, i.e. as straight as possible and without obstacles if possible. This simple strategy is a way to take into account previous experiences of the robotics system to exploit them in future navigation tasks [42].

7.2. Peripheral avoidance

The task function to circumvent an obstacle is defined as:

$$e_o = d_o - d_s \in \mathbb{R}, \tag{9}$$

where

$$d_o = ||\boldsymbol{x}_r - \boldsymbol{x}_o||, \tag{10}$$

with d_s being the security distance around the obstacle. The vectors $\boldsymbol{x}_r = [x_r \; y_r]^T \in \mathbb{R}^2$ and $\boldsymbol{x}_o = [x_o \; y_o]^T \in \mathbb{R}^2$ correspond to the relative robot and obstacle positions with respect to the current frame $\mathcal{C}$. By differentiating the task error (9) with respect to the vector $\boldsymbol{x}_r - \boldsymbol{x}_o$, we obtain its gradient:

$$\boldsymbol{g}_o = \left(\frac{\boldsymbol{x}_r - \boldsymbol{x}_o}{d_o}\right)^T \in \mathbb{R}^{1\times 2}. \tag{11}$$

Then, by defining a common reference frame to express the obstacle avoidance and the visual tasks, e.g. the corresponding key image frame $\mathcal{C}_k^*$, we obtain:

$$\dot{e}_o = \boldsymbol{J}_o \boldsymbol{\nu}_c, \tag{12}$$

where $\boldsymbol{J}_o$ represents the obstacle avoidance Jacobian:

$$\boldsymbol{J}_o = \begin{pmatrix} \boldsymbol{g}_o & \mathbf{0}_{1\times 4} \end{pmatrix} \in \mathbb{R}^{1\times 6} \tag{13}$$

and its corresponding null-space projector is given by:

$$\boldsymbol{Q}_o = \boldsymbol{I} - \boldsymbol{J}_o^T \boldsymbol{J}_o \in \mathbb{R}^{6\times 6}. \tag{14}$$

By activating the task (12) at the first hierarchical level, an abrupt change in the reference velocity of the robot's CoM occurs (see [43] and [7]). This is a well-known problem in the visual control community where some solutions have been suggested, e.g. in [44] and [15].

In particular, we adopted the intermediate value strategy introduced in [45]. By handling smooth task transitions, the camera twist is computed as:

$$\begin{aligned} \boldsymbol{\nu}_c &= h(t)\left(\boldsymbol{\nu}'_o + \boldsymbol{\nu}'_{\mathrm{v|o}}\right), \\ \boldsymbol{\nu}'_o &= \boldsymbol{J}_o^T \dot{e}'_o, \\ \boldsymbol{\nu}'_{\mathrm{v|o}} &= \boldsymbol{Q}_o\left(-\lambda_{\mathrm{v}}\mathbf{s} - \boldsymbol{\nu}'_o\right), \end{aligned} \tag{15}$$

where

$$\dot{e}'_o = -\lambda_o h_{\mathrm{t}}(t) e_o - \left(1 - h_{\mathrm{t}}(t)\right) \boldsymbol{J}_o \lambda_{\mathrm{v}} \mathbf{s} \tag{16}$$

encodes the intermediate desired values. The twist to circumvent the obstacle is $\boldsymbol{\nu}'_o$, and $\boldsymbol{\nu}'_{\mathrm{v|o}}$ is the resulting twist for achieving the visual task without perturbing the execution of the obstacle avoidance task. The gains λ_o and λ_{v} are related to the obstacle and visual tasks, respectively. The smooth transition function $h_{\mathrm{t}}(t)$ varies within the interval $0 \leq h_{\mathrm{t}}(t) \leq 1$. Thus, $h_{\mathrm{t}}(t) = 0$ implies that $e_o > 0$, and the obstacle avoidance task is not active. Otherwise, $e_o \leq 0$, and the function $h_{\mathrm{t}}(t)$ smoothly increases up to its maximum value while the obstacle avoidance task becomes active. The stability proof of the control law (15) is reported in [7].

7.3. Step over the obstacle

This obstacle avoidance behavior is achieved by solving several hierarchical quadratic programs (HQP) at kinematic level. Each HQP is formulated as in [46]:

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}_i, \boldsymbol{w}_i} \ \tfrac{1}{2} \parallel \boldsymbol{w}_i \parallel^2 \\ &\text{s.t.} \ \boldsymbol{b}_i^l \leq \boldsymbol{J}_i \dot{\boldsymbol{q}}_i - \boldsymbol{w}_i \leq \boldsymbol{b}_i^u \\ &\qquad \overline{\boldsymbol{b}}_i^l \leq \ \overline{\boldsymbol{J}}_{i-1} \dot{\boldsymbol{q}}_i \ \ \leq \overline{\boldsymbol{b}}_i^u \end{aligned} \tag{17}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the humanoid's joint velocity vector, $\boldsymbol{w} \in \mathbb{R}^m$ is a vector of slack variables used to relax the infeasible constraints at hierarchical level i, $\boldsymbol{J}_i$ is the task Jacobian, $\boldsymbol{b}_i^l$ and $\boldsymbol{b}_i^u$ represent the corresponding lower and upper bounds, respectively. Note that the solution of higher hierarchical tasks is maintained by adding them in:

$$\overline{\boldsymbol{J}}_i = \begin{pmatrix} \overline{\boldsymbol{J}}_{i-1} \\ \boldsymbol{J}_i \end{pmatrix}, \quad \overline{\boldsymbol{b}}_i(\boldsymbol{q}) = \begin{pmatrix} \overline{\boldsymbol{b}}_{i-1} \\ \boldsymbol{J}_i \dot{\boldsymbol{q}}_i^* + \boldsymbol{w}_i^* \end{pmatrix} \tag{18}$$

where $\dot{\boldsymbol{q}}_i^*$ is the optimal solution at i. The active inequalities at i are also added to (18). It is important to mention that a dedicated HQP can be employed to dramatically reduce the computational cost [41].

The step over behavior corresponds to a quasi-static motion since the projection of the humanoid's CoM on the floor is constrained to move inside the support polygon defined by the boundary of the contact area. In other words, the center of pressure criterion is not considered for this task. The whole-body motion is composed by a sequence of several ordered arrays of hierarchical tasks. Each array contains the visual task at the lowest hierarchy.

In Algorithm 3, MotionTasks is an input to the step over obstacle strategy. In particular, it refers to the ordered array of hierarchical tasks to maintain the contact of the humanoid feet on the floor, to generate the motion of the CoM inside the support polygon for either single and double support stages, and to track the stepping trajectory. In addition, several inequality constraints are also imposed for joint position and velocity limits as well as for avoiding self collisions (e.g., the humanoid hands are constrained to move within a safe region of the task space). Each motion task is expressed as a linear differential system to be solved by means of (17).

8. EXPERIMENTAL EVALUATION

We implemented the proposed navigation scheme in a NAO humanoid robot. The top camera of the NAO's head was used as main sensor in the experiments and an RGB-D camera was mounted on the robot to detect obstacles. We divided the experimental evaluation in two parts, one related to the proposed appearance-based localization and the other related to the integration of the whole humanoid navigation scheme, i.e., localization and planning, path following with obstacle avoidance and graph updating.

For all the experimental evaluations the images captured by the humanoid were obtained with a resolution of 640 $\times$ 480 pixels. The image features were acquired as follows: first, a corner detector based on [47] was used, which is implemented in the function `goodFeaturesToTrack` of the OpenCV library. Then, we assigned a SURF descriptor [48] to each detected point. To estimate the homography model, a robust matcher based on RANSAC matched all the points between the current image and the corresponding key image. This matching procedure was used in the localization and path following stages to initialize the points to be tracked. The `HLM` function of ViSP library [49] was used to compute the homography for planar and non-planar scenes [32]. All the poses of the robot during the experiments were captured with an Optitrack System to obtain ground truth of the navigation scheme.

8.1. Localization

In this section, we present an evaluation of the localization method detailed in Algorithm 2.

8.1.1. Boundary of the initialization around a key image

First, we made an evaluation of the boundaries around a key image where the robot can be initialized, i.e., the region where the localization algorithm works effectively. For this evaluation, we proposed two experimental settings for near and far scenes, respectively. To obtain the working boundaries, we varied the robot placement (position and orientation) with respect to a key image as shown in Fig. 4 to have what we call evaluation poses.

We defined two measurements for estimating the working boundary. One is the number of points matched between the key image and the images taken at the evaluation poses. The second measurement is the dispersion or distribution of the matched points in the image plane; the greater the dispersion, the greater the probability of good localization and navigation. To calculate a degree of dispersion, we build a symmetric matrix $\mathbf{U}$ that contains the set of matched points, previously centered around their mean values $(\overline{u}, \overline{v})$. The eigenvalues of the symmetric matrix $\mathbf{U}^T\mathbf{U}$ (calculated through its determinant) span a quadratic form whose

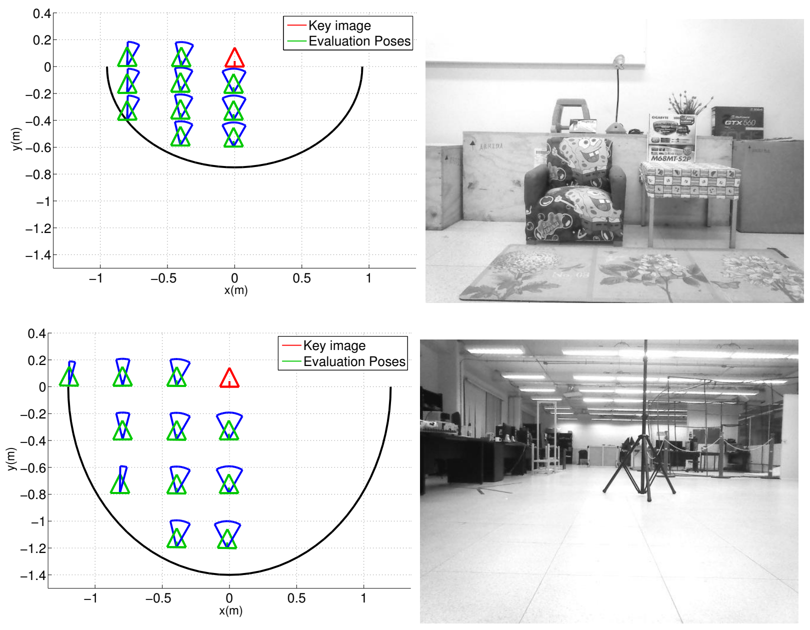

Figure 4: **Boundary evaluation around a key image**. The red triangles represent the poses of the robot where the key image was taken. **Above.** The boundary and the key image for a near scene. **Below.** The boundary and the key image for a far scene. The green triangles are some samples of the evaluation poses. The blue cones represent the range of orientations for these samples, they varied $-30°$, $-15°$, $0°$, $15°$ and $30°$.

area with respect to the total number of pixels can be related to the dispersion d of the set as follows:

$$d = \frac{\pi\sqrt{det(\mathbf{U}^T\mathbf{U})}}{4u_c v_c}. \tag{19}$$

where (u_c, v_c) are the coordinates of the image center.

Due to the homography computation for non-planar scenes, we know that the localization algorithm and the control law need at least $\mu > 8$ matches to work effectively. Additionally, we set a minimum dispersion $d = 20$ (defined experimentally) to estimate the working regions shown in Fig. 4. The black arc represents the region where the robot can be correctly initialized with respect to the key image. As expected, when the scene is far, the region in which the robot can start to navigate is bigger than when the scene is near. Being conservative, we concluded that, in general, the boundary of the region in which the robot is adequately initialized corresponds to a semicircle of radius 0.75 meters behind a key image, considering forward motion direction.

In these results and in general for the experimental evaluation, we request 1000 feature points to the detector `goodFeaturesToTrack` with the additional requirement of having a minimum distance of 31 pixeles between points (`minDistance` parameter of the function `goodFeaturesToTrack`) for images of 640x480. According to our results, we consider that this strategy for points detection has been enough to achieve a correct distribution of point features and consequently to have a good boundary of the initialization. However, a strategy like the one in [34] to force a uniform layout of points over the frames could also be used.

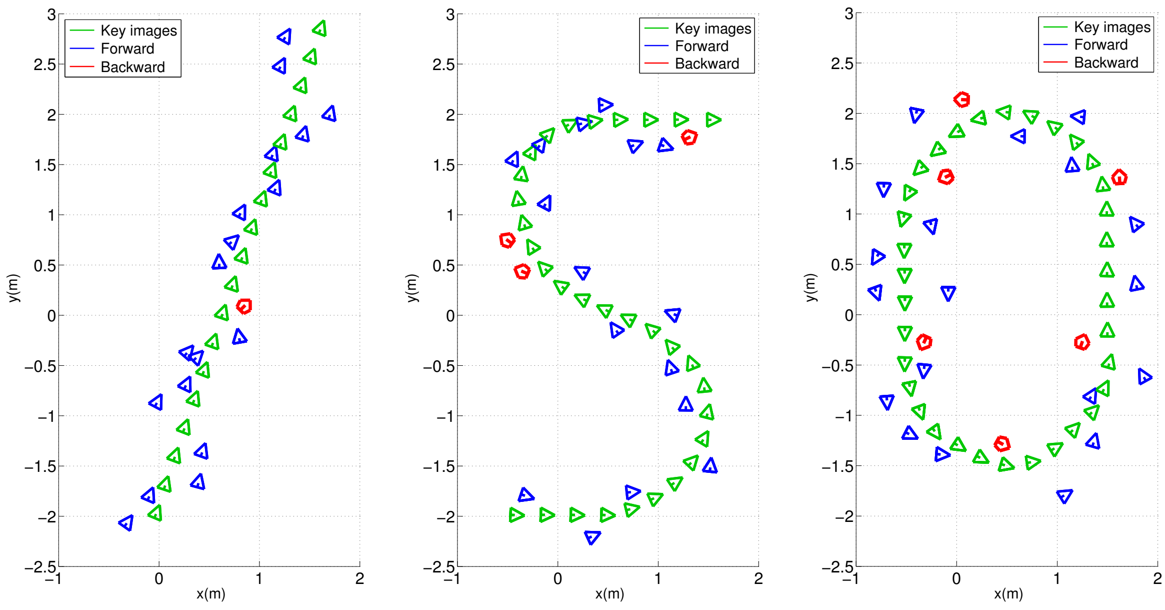

Figure 5: **Results of localization in one branch.** We used three visual paths: a straight line, a S-like path and an elliptic path. The green triangles represent the robot's poses where key images were taken. The blue triangles represent the evaluation poses where the localization found forward images. The red pentagons represent results of localization for backward images. All this information was captured with the Optitrack System.

8.1.2. Localization in one branch

First, we present the evaluation of the localization Algorithm 2 for the case when there is only one branch or visual path. Thus, the localization process is performed by the lines 19 to 23 in Algorithm 2, which selects the closest key image $\mathcal{I}_1^*$ to the current image $\mathcal{I}$.

We applied Algorithm 2 to each path shown in Fig. 5, but performing exhaustive comparisons in the localization process (line 1 of Algorithm 2 is not used). The results are shown in Table 1. We classified the results in two categories, the case when the closest key image was found ahead the robot, denoted as "Forward" and the case when the closest key image was found "Backward" the robot. We observed that the localization algorithm always found a solution. In particular, the resulting images were forward in more than 80% of the trials. Clearly, it is preferred to perform a forward navigation from the beginning of the autonomous motion. However, in 16% of the cases the key images were found behind the robot.

Table 1: Evaluation of the localization for single paths.

Path	Forward	Backward
Line	18	1
"S"	16	3
Ellipse	19	6
Total (%)	53 (84)	10 (16)

8.1.3. Localization aided by planning

In these experiments, the localization was carried out with the graph depicted in Fig. 6. In this case, the two candidate images $(\mathcal{I}_{\pm1}^*, \mathcal{I}_{\pm2}^*)$ referred in line 18 of Algorithm 2 might be

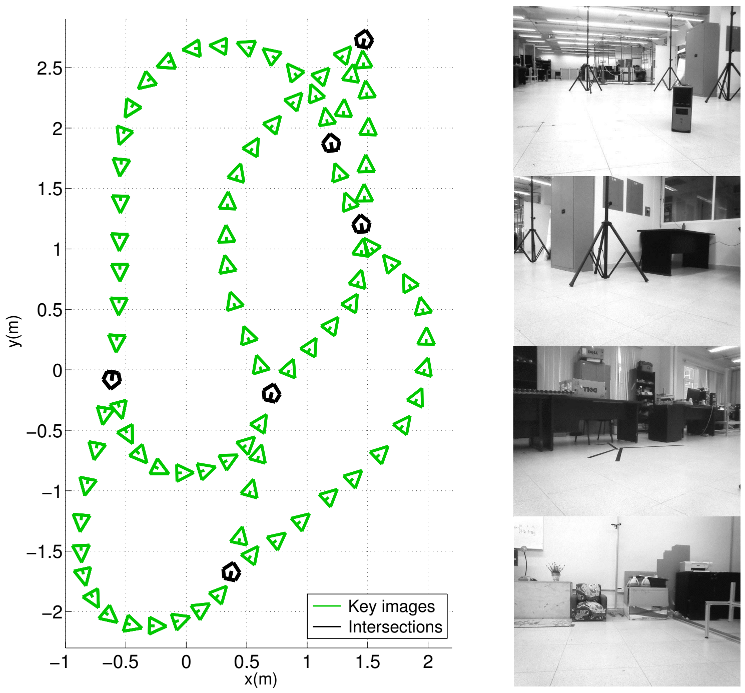

Figure 6: **Navigation graph and examples of key images.** The green triangles represent the pose of the robot where each key image (node) was taken. The black pentagons are intersection nodes that connect the branches of the graph. Some images that form the VM are shown at the right.

in the same branch or in different branches of the graph. The VM contains 89 key images (see Fig. 6). Some branches of the VM are connected by "Intersection" nodes. For the planning algorithm, we decided to use the Dijkstra's algorithm in **getShortestPath** to obtain the minimum length path.

The localization requires the homography matrix decomposition to obtain the scaled translation vector $\mathbf{t}$. An efficient algorithm to decompose $\mathbf{H}$ is suggested in [50]. Such kind of decomposition generates two geometrically valid solutions where only one of them is physically admissible. The correct solution can be selected by taking the normal vector whose third value (n_z) is the largest. Although the vector $\mathbf{t}$ is scaled, it gives the direction and a notion of distance of the key images with respect to the current image if the virtual plane is fixed as described in Section 5.

In Fig. 7 we present three cases of localization and planning. The first case is shown in Fig. 7 (left) where the robot starts near one branch (red triangle). The two candidate key images $\mathcal{I}^*_{\pm 1}$ and $\mathcal{I}^*_{\pm 2}$ (orange circle and gray square) are in the same branch, and the localization algorithm selects the closest key image $\mathcal{I}^*_1$ (orange circle). Once the robot is localized, **getShortestPath** finds the minimum length path $\mathbf{I}^*$ to the target image $\mathcal{I}^*_n$ (magenta triangle at top). The second and third cases are shown in Fig. 7 (center and right). The robot starts between two branches of the graph (red triangle) such that the two candidate key images belong to different branches (orange circle and gray square). In these cases, the localization algorithm is aided by the path planner to select the most similar key image while taking into account the shortest path to the target image. As it can be seen in Fig. 7 (center), the shortest path to the target image (magenta triangle) corresponds to the key image on the right (orange circle), but if the target image changes, as it is observed in Fig. 7

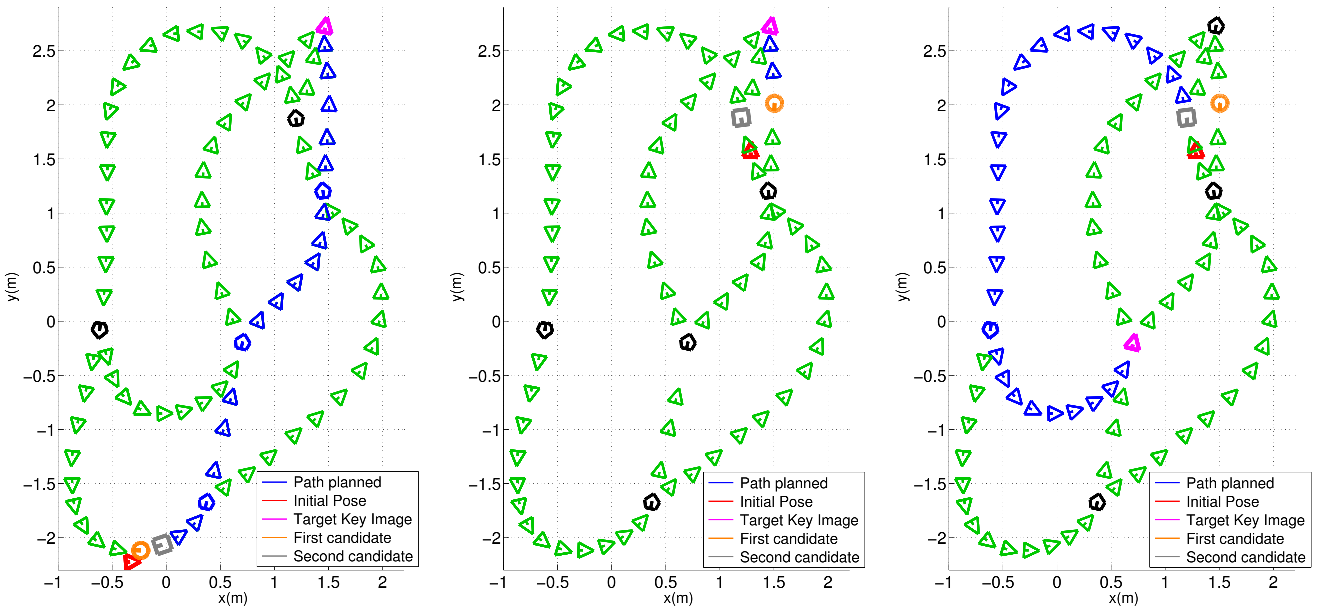

Figure 7: **Evaluation of the localization and planning.** Left: Case when the two candidate images are in the same branch of the graph. Center and right: Cases when the two candidate images are in different branches.

(right), the algorithm selects the key image on the left (gray square). Hence, the localization avoids long paths for initializing the robot navigation.

We evaluated the CPU time dedicated to solve the localization by comparing the divide-and-conquer strategy using line 1 of Algorithm 2 against the exhaustive search for the same graph of Fig. 6. This evaluation was done offline using a laptop with CPU Intel Core i7 of 2.20 GHz with 8.00 GB in RAM. The results are shown in Table 2 for 18 trials with different initialization images. Since the termination condition of the function **findNodesNeighborhood** depends on μ, we decided to evaluate the localization with 8 and 16 matches. By setting a higher value of μ, the localization retrieved better key images in terms of common visual information. Also, the computation time does not suffer an important increment compared to the exhaustive procedure.

Table 2: CPU time to solve the robot localization.

Method	Compared nodes	Time (s)
Division $\mu = 8$	22.6 (Average)	1.94 (Average)
Division $\mu = 16$	26.6 (Average)	2.27 (Average)
Exhaustive	89	7.3

8.2. Visual navigation with obstacle avoidance

We evaluated the performance of the visual path following scheme described in Algorithm 3. Notice that the input of this stage is the sequence of key images of the planned visual path. For this experiment, we used the initialization procedure of image features described at

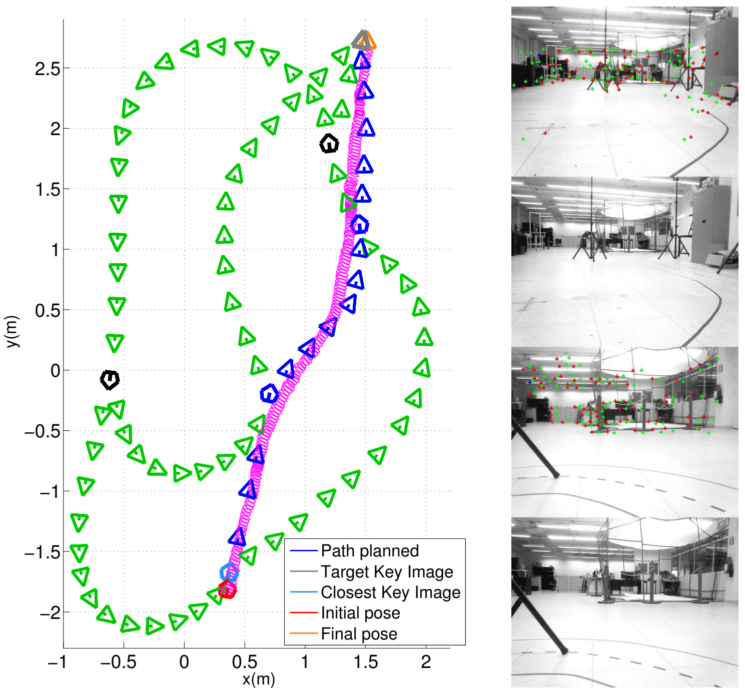

Figure 8: **Performance of the autonomous robot navigation.** The magenta line shows the motion of the robot during the navigation. As it can be seen, the robot gets close to the target key image (gray triangle) with an error < 10cm. The images on the right from top to bottom are: the current image $\mathcal{I}$ at the initial pose (red pentagon), the closest key image ${}^{p}\mathcal{I}_1^*$ (light blue pentagon), the current image at the final pose (orange triangle) and the target key image $\mathcal{I}^*$ (gray triangle).

the beginning of this section. During the locomotion, we used a tracking algorithm based on a sparse iterative version of the Lucas-Kanade optical flow in pyramids, implemented in the function `calcOpticalFlowPyrLK` of OpenCV. We evaluated experimentally the performance of the visual tracker with the NAO robot subject to the effect of the sway motion generated by the robot's locomotion. This issue has been addressed in the literature by canceling the oscillations in the feedback error [5], or by filtering the measurements [18]. In our case, we do not focus in a way to cancel the sway motion, we just tried to mitigate its effect by ensuring a good tracking of point features in spite of this unavoidable robot motion. This was achieved by tuning the points tracker setting a number of 3 pyramids and an appropriate size window of 31 pixels.

Since the robot's translation and rotation are decoupled, different control gains were used in the control law (15): $\lambda_{\mathrm{v}} = 0.11$ for frontal translation, $\lambda_{\mathrm{v}} = 0.055$ for lateral translation, $\lambda_{\mathrm{v}} = 0.6$ for rotation and $\lambda_o = 0.6$. We set the transition time interval for $h_t(t)$ in (16) as $t_{t,f} - t_{t,0} = 10$ seconds for the obstacle peripheral avoidance. The duration of the transition function $h(t)$ was set experimentally to $t_f - t_0 = 3.5$ seconds to mitigate the discontinuities of walking velocities when the robot switches to the next key image for computing the visual error. Also, we defined experimentally $h_0 = 0.2$ to keep a minimal walking velocity along the robot navigation. It is worth emphasizing that a small threshold T_ε for switching to the next key image was not always the best option to obtain a natural robot motion during navigation. Indeed, we have achieved a good behavior by fixing $T_\varepsilon = 18$ pixels.

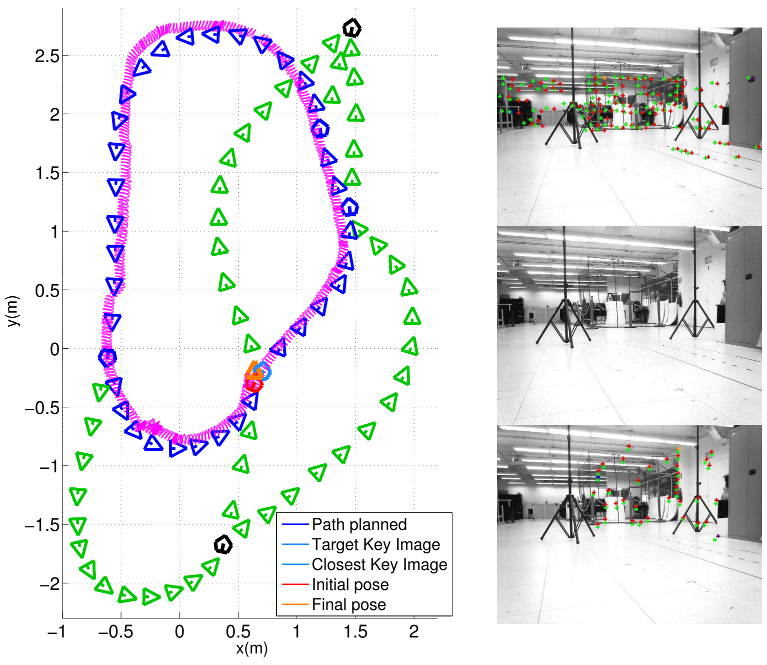

Figure 9: **Closed-loop path.** The magenta line shows the motion of the robot during the navigation. In this case, the target key image $\mathcal{I}_n^*$ and the most similar key image $\mathcal{I}_1^*$ are depicted by the same light blue pentagon. The images on the right from top to bottom are: the current image $\mathcal{I}$ at the initial pose (red pentagon), the target key image $\mathcal{I}_n^*$ (light blue pentagon) and the current image at the final pose (orange triangle).

We show the behavior of the robot for the whole navigation scheme according to Algorithm 1. Fig. 8 shows the performance of the whole navigation scheme using the humanoid robot NAO. The red pentagon depicts the initial pose of the robot, the light blue pentagon is the most similar key image that Algorithm 2 returned. The target key image is the gray triangle, and the orange triangle is the final pose of the robot during the experiment. The planned visual path contained 18 key images over a total distance of 4.7m.

8.2.1. Closed-loop path

The behavior of the robot for a navigation in a loop is shown in Fig. 9, i.e., the target key image $\mathcal{I}_n^*$ is the most similar key image $\mathcal{I}_1^*$ (both are shown as the light blue pentagon) given by the localization algorithm. Therefore, the robot starts (red pentagon) and ends (orange triangle) near the location of the target image. The blue triangles refer to the path of 38 images followed by the robot over a distance of approximately 9m. It can be seen that the proposed scheme effectively guided the robot to follow a path in closed-loop.

8.2.2. Obstacle avoidance and graph updating

To evaluate the obstacle avoidance and graph updating mechanisms, we proposed four experiments with four different paths for which static obstacles appeared during the execution of the visual path following stage.

Fig. 10 shows the detection of two types of obstacles with distance filters on images obtained from the RGB-D sensor. To distinguish between thin (peripheral avoidance) and

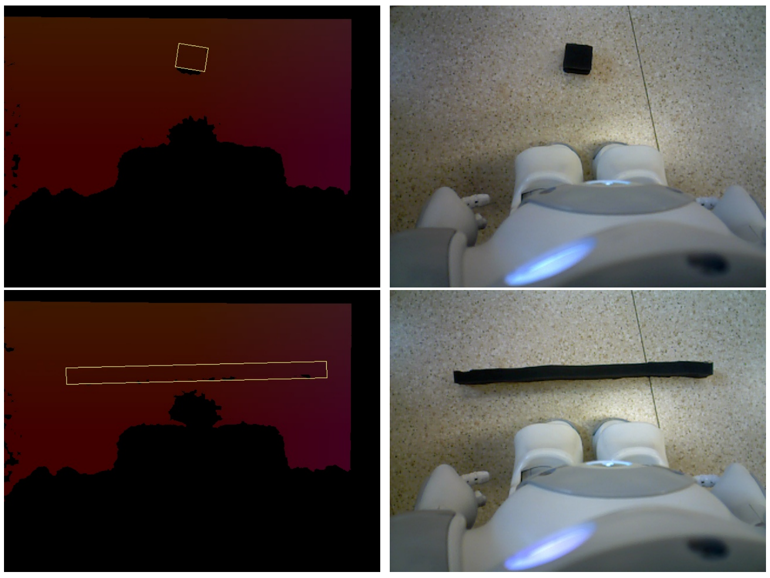

Figure 10: **Detection of the two types of obstacles (short and long) with the ASUS Xtion RGB-D sensor. Top:** small thin obstacle (the robot is able to surround it). **Bottom:** wide obstacle (the robot is not able to surround it, but it is able to step over, for example a fence).

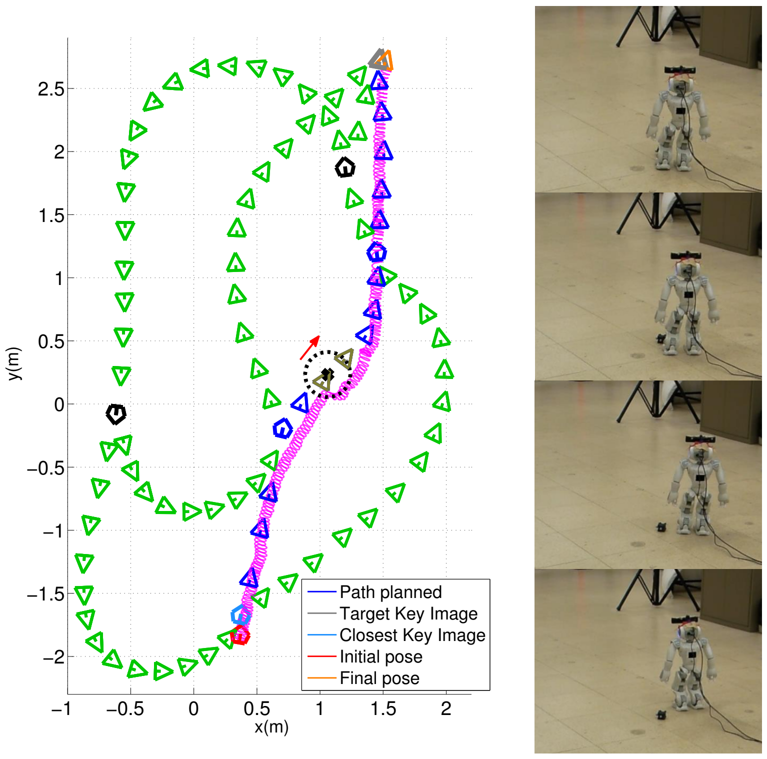

Figure 11: **First navigation with obstacles: peripheral avoidance.** The magenta line shows the motion of the robot during navigation with peripheral avoidance. The dotted black circle represents the security distance d_s that activates the avoidance. The red arrow represents the edge $\mathcal{E}_i$ where the robot located the obstacle in memory. The images on the right show some snapshots of the robot during the peripheral avoidance.

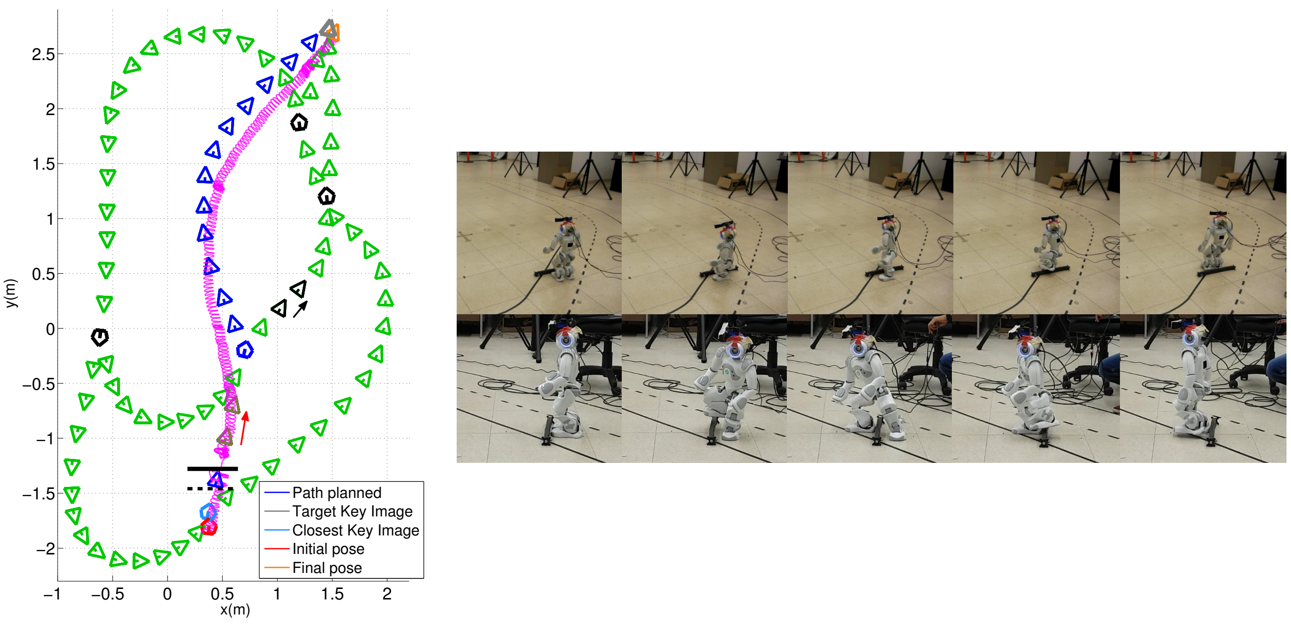

Figure 12: **Second navigation with obstacles: stepping over avoidance.** The magenta line shows the motion of the robot during navigation with stepping over avoidance. The solid and dotted black lines represent the position of the obstacle and the security distance d_s, respectively. The red arrow shows the edge $\mathcal{E}_i$ where the robot located the large obstacle during the experiment. The black arrow represents the weighted edge $W\mathcal{E}_i$ where the robot located the short obstacle in Fig. 11. The images at the bottom show some snapshots of the robot during the stepping over avoidance.

wide (stepping over) obstacles, we defined $A = 0.1$m in line 13 of Algorithm 3. Since our aim was to show the performance of the robot navigation, the shape of the obstacles as well as the image processing method for object segmentation are rather simple.

Fig. 11 shows the first experiment of navigation with obstacle avoidance, where the blue triangles refer to the planned path toward the target key image (gray triangle). In this case, the obstacle length was $L_o < A$, therefore the robot surrounded it. The black dotted circle represents the security distance $d_s = 0.13$m that, if it is violated then the smooth transition control of equation (15) is activated using $h_t(t) \to 1$. The brown triangles indicate the nodes associated to the edge $\mathcal{E}_i$ (represented by the red arrow) where the robot located the obstacle during the experiment, i.e. the edge that will be updated with a large weight for the next navigation.

Fig. 12 illustrates the second experiment performed with the updated graph. Although the target key image (gray triangle) was the same as in the experiment of Fig. 11, the resulting path (blue triangles) was different. This is due to the higher cost encoded in the previously updated edge where the obstacle avoidance occurred (see Fig. 11). Concerning the second experiment, the obstacle avoidance condition $L_o > A$ was true. Hence, the robot stepped over the obstacle. The dotted black line in Fig. 12 represents $d_s = 0.13$m, which was violated according to Algorithm 3, and the **stepOverObstacle** function handled the robot motion execution. The red arrow and the brown triangles show the localization of the obstacle during the experiment. In this case, the corresponding edge was also updated for future navigation. As it can be observed in Fig. 12, the edge detection did not reflect the proper location of the obstacle because the localization process uses the topology of the graph, which is not spatial. However, according to our experiments, the localization of the obstacles has an error of no more than one neighboring edge, which is sufficient to detect and to avoid paths with obstacles. Thus, the information of the localized obstacles is used to update the graph and to avoid planning paths containing obstacles as much as possible in future navigations.

We have included a *video as a multimedia extension* of the paper, in which most of the experiments reported herein are shown during their execution. The video also shows experiments in cluttered and corridor-like environments, where the robot performs lateral motion, since in those cases, it is demanded by the VM. These experiments demonstrate that the proposed scheme allows to consider the holonomic nature of humanoid robots, providing a versatile navigation solution.

9. CONCLUSIONS

In this paper, we have proposed a vision-based navigation scheme for humanoid robots that relies on a topological representation of the environment known as visual memory. Such environment model is constructed in a human-guided teaching phase, in which a set of key images are captured and organized as a directed graph. We have shown that this graph is enough to qualitatively solve the robot localization given the current image from the robot's camera. The proposed appearance-based localization method finds the most similar key image with respect to the current image in terms of common visual information with sufficient accuracy. Once the robot is localized and given a desired target key image associated to the desired robot location, the visual path planner returns the sequence of key images to reach the desired location. The experiments have shown that the humanoid is able to follow the planned visual path. The visual-servo controller, that only relies on 2D information, drives the robot to a vicinity of the location associated to a target key image, being sufficient for a navigation task, rather than reaching the target location or following the path with high accuracy.

It is clear that in static scenarios, the navigation based on a visual memory naturally generates obstacle free paths since obstacles are taken into account in the initial supervised navigation stage. However, it is important to deal with changes in the visual memory due to obstacles that were not initially present. Thus, we have introduced an obstacle avoidance task for stepping over and circumvent obstacles. We have shown that the humanoid robot is able to avoid obstacles while keeping the performance of the visual path following. This has been handled by means of the task-based control framework, in which the transitions of successive and hierarchical tasks are performed smoothly to keep the balance of the humanoid robot during navigation. In addition, we have shown that the humanoid localization and visual path following can be completely based on 2D information by taking advantage of the homography for planar and non planar scenes. Moreover, the proposed navigation scheme can be extended to rely on other geometric constraints like the epipolar geometry or the trifocal tensor, since they can provide the estimated translation and rotation required by the controller. In that case, it might be useful to combine models. Although in our results the obstacle avoidance strategy relies on depth measurements of an RGB-D sensor, this can be tackled in the future by using only the monocular camera.

An experimental evaluation of the different stages of the navigation as well as its integration have been reported in this paper using the NAO platform. We have shown experimentally that our navigation scheme works effectively in different indoor environments like corridors, uncluttered or cluttered environments. We have taken advantage of the motion capabilities of humanoid robots, since no motion constraints are imposed by the navigation scheme, allowing the robot to walk in any direction as required by the visual path and the obstacle avoidance strategy.

As future work, we plan to extend the obstacle avoidance strategy to deal with moving obstacles and to deal with huge visual memories. Additionally, since there exist several visual SLAM techniques based on keyframes, one could try to combine the proposed navigation scheme with those methods in order to take advantages of both 2D and 3D worlds.

References

[1] A. H. Javadi, B. Emo, L. Howard, F. Zisch, Y. Yu, R. Knight, J. P. Silva, H. J. Spiers, Hippocampal and prefrontal processing of network topology to simulate the future, Nature Communications.

[2] Y. Matsumoto, M. Inaba, H. Inoue, Visual navigation using view-sequenced route representation, in: IEEE Int. Conf. on Robotics and Automation, 1996, pp. 83–88.

[3] J. Courbon, Y. Mezouar, P. Martinet, Indoor navigation of a non-holonomic mobile robot using a visual memory, Autonomous Robots 2008 (25) (2008) 253–266.

[4] J. Courbon, Y. Mezouar, P. Martinet, Autonomous navigation of vehicles from a visual memory using a generic camera model, IEEE Trans. on Intelligent Transportation Systems 10 (3) (2009) 392–402.

[5] C. Dune, A. Herdt, O. Stasse, P. B. Wieber, K. Yokoi, E. Yoshida, Cancelling the sway motion of dynamic walking in visual servoing, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2010, pp. 3175–3180.

[6] M. García, O. Stasse, J. B. Hayet, C. Dune, C. Esteves, J. P. Laumond, Vision-guided motion primitives for humanoid reactive walking: decoupled versus coupled approaches, The Int. Journal of Robotics Research 34 (4–5) (2014) 402–419.

[7] J. Delfin, H. M. Becerra, G. Arechavaleta, Visual servo walking control for humanoids with finite-time convergence and smooth robot velocities, Int. Journal of Control 89 (7) (2016) 1342–1358.

[8] G. Lopez-Nicolas, N. R. Gans, S. Bhattacharya, C. Sagues, J. J. Guerrero, S. Hutchinson, Homography-based control scheme for mobile robots with nonholonomic and field-of-view constraints, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 40 (4) (2010) 1115–1127.

[9] F. Chaumette, S. Hutchinson, Visual servo control. Part I: Basic approaches, IEEE Robotics and Automation Magazine 13 (4) (2006) 82–90.

[10] J. Ido, Y. Shimizu, Y. Matsumoto, T. Ogasawara, Indoor navigation for a humanoid robot using a view sequence, The Int. Journal of Robotics Research 28 (2) (2009) 315–325.

[11] A. Diosi, S. Segvic, A. Remazeilles, F. Chaumette, Experimental evaluation of autonomous driving based on visual memory and image-based visual servoing, IEEE Trans. on Intelligent Transportation Systems 12 (3) (2011) 870–833.

[12] H. M. Becerra, C. Sagüés, Y. Mezouar, J. B. Hayet, Visual navigation of wheeled mobile robots using direct feedback of a geometric constraint, Autonomous Robots 37 (2) (2014) 137–156.

[13] J. Delfin, H. M. Becerra, 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 Robots, 2014, pp. 354–359.

[14] J. Delfin, H. M. Becerra, G. Arechavaleta, Humanoid localization and navigation using a visual memory, in: IEEE-RAS Int. Conf. on Humanoid Robots, 2016, pp. 75–80.

[15] A. Cherubini, F. Chaumette, Visual navigation of a mobile robot with laser-based collision avoidance, The Int. Journal of Robotics Research 32 (2) (2013) 189–205.

[16] N. Mansard, O. Stasse, F. Chaumette, K. Yokoi, Visually-guided grasping while walking on a humanoid robot, in: IEEE Int. Conf. on Robotics and Automation, 2007, pp. 3041–3047.

[17] A. Herdt, H. Diedam, P. B. Wieber, D. Dimitrov, K. Mombaur, M. Diehl, Online walking motion generation with automatic footstep placement, Advanced Robotics 24 (5-6) (2010) 719–737.

[18] G. Oriolo, A. Paolillo, L. Rosa, M. Vendittelli, Vision-based trajectory control for humanoid navigation, in: IEEE-RAS Int. Conf. on Humanoid Robots, 2013, pp. 118–123.

[19] A. Paolillo, A. Faragasso, G. Oriolo, M. Vendittelli, Vision-based maze navigation for humanoid robots, Autonomous Robots 41 (2) (2017) 293–309.

[20] J. H. Hayet, C. Esteves, G. Arechavaleta, O. Stasse, E. Yoshida, Humanoid locomotion planning for visually-guided tasks, Int. Journal of Humanoid Robots 9 (2) (2012) 26.

[21] G. Oriolo, A. Paolillo, L. Rosa, M. Vendittelli, Humanoid odometric localization integrating kinematics, inertial and visual information, Autonomous Robots 40 (5) (2016) 867–879.

[22] L. George, A. Mazel, Humanoid robot indoor navigation based on 2D bar codes: Application to the NAO robot, in: IEEE-RAS Int. Conf. on Humanoid Robots, 2013, pp. 329–335.

[23] R. Cupec, G. Schmidt, O. Lorch, Vision-guided walking in a structured indoor scenario, Automatika 46 (1-2) (2005) 49–57.

[24] D. Maier, C. Stachniss, M. Bennewitz, Vision-based humanoid navigation using self-supervised obstacle detection, Int. Journal of Humanoid Robotics 10 (2) (2013) 1–28.

[25] O. Stasse, B. Verrelst, B. Vanderborght, K. Yokoi, Strategies for humanoid robots to dynamically walk over large obstacles, IEEE Trans. on Robotics 25 (4) (2009) 960–967.

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

[27] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, P. Souères, A reactive walking pattern generator based on nonlinear model predictive control, IEEE Robotics and Automation Letters 2 (1) (2017) 10–17.

[28] M. Ferro, A. Paolillo, A. Cherubini, M. Vendittelli, Omnidirectional humanoid navigation in cluttered environments based on optical flow information, in: IEEE-RAS Int. Conf. on Humanoid Robots, 2016, pp. 75–80.

[29] A. Rioux, W. Suleiman, Autonomous slam based humanoid navigation in a cluttered environment while transporting a heavy load, Robotics and Autonomous Systems 99 (2018) 50 – 62.

[30] J. Stalbaum, J.-B. Song, Keyframe and inlier selection for visual slam, in: Int. Conf. on Ubiquitous Robots and Ambient Intelligence, 2013, pp. 391–396.

[31] G. Guan, Z. Wang, S. Lu, J. Da-Deng, D-Dagan-Feng, Keypoint-based keyframe selection, IEEE Trans. on Circuits and Systems for Video Technology 23 (4) (2013) 729–734.

[32] E. Malis, F. Chaumette, S. Boudet, 2 1/2 Visual servoing with respect to unknown objects through a new estimation scheme of camera displacement, Int. Journal of Computer Vision 37 (1) (2000) 79–97.

[33] R. I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, 2004.

[34] R. Mur-Artal, J. M. M. Montiel, J. D. Tardos, ORB-SLAM: a versatile and accurate monocular SLAM system, IEEE Transactions on Robotics 31 (5) (2015) 1147–1163.

[35] T. H. Cormen, Introduction to algorithms, MIT press, 2009.

[36] I. Kostavelis, A. Gasteratos, Learning spatially semantic representations for cognitive robot navigation, Robotics and Autonomous Systems 61 (12) (2013) 1460–1475.

[37] D. Gálvez-López, J. D. Tardos, Bags of binary words for fast place recognition in image sequences, IEEE Transactions on Robotics 28 (5) (2012) 1188–1197.

[38] N. G. Aldana-Murillo, J. B. Hayet, H. M. Becerra, Comparison of local descriptors for humanoid robots localization using a visual bag of words approach, Intelligent Automation and Soft Computing (2017) 1–11.

[39] A. Babenko, V. Lempitsky, The inverted multi-index, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (6) (2015) 1247–1260.

[40] M. Muja, D. G. Lowe, Scalable nearest neighbor algorithms for high dimensional data, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (11) (2014) 2227–2240.

[41] A. Escande, N. Mansard, P. B. Wieber, Hierarchical quadratic programming: Fast online humanoid-robot motion generation, The Int. Journal of Robotics Research 33 (7) (2014) 1006–1028.

[42] L. Nardi, C. Stachniss, User preferred behaviors for robot navigation exploiting previous experiences, Robotics and Autonomous Systems 97 (2017) 204 – 216.

[43] F. Keith, P. B. Wieber, N. Mansard, A. Kheddar, Analysis of the discontinuities in prioritized task-space control under discrete task scheduling operations, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2011, pp. 3887–3892.

[44] N. Mansard, A. Remazeilles, F. Chaumette, Continuity of varying-feature-set control laws, IEEE Trans. on Automatic Control 54 (11) (2009) 2493–2505.

[45] J. Lee, N. Mansard, J. Park, Intermediate desired value approach for task transition of robots in kinematic control, IEEE Trans. on Robotics 28 (6) (2012) 1260–1277.

[46] O. Kanoun, F. Lamiraux, P.-B. Wieber, Kinematic Control of Redundant Manipulators: Generalizing the task-priority framework to inequality task, IEEE Trans. on Robotics 27 (4) (2011) 785–792.

[47] J. Shi, C. Tomasi, Good features to track, in: IEEE Conf. on Computer Vision and Pattern Recognition, 1994, pp. 593–600.

[48] H. Bay, T. Tuytelaars, L. V. Gool, SURF: Speeded up robust features, in: European Conf. on Computer Vision, 2006, pp. 404–417.

[49] E. Marchand, F. Spindler, F. Chaumette, Visp for visual servoing: a generic software platform with a wide class of robot control skills, IEEE Robotics and Automation Magazine 12 (4) (2005) 40–52.

[50] B. Triggs, Autocalibration from planar scenes, in: H. Burkhardt, B. Neumann (Eds.), Computer Vision - ECCV'98, Vol. 1406 of LNCS, Springer Berlin Heidelberg, 1998, pp. 89–105.