Humanoid Localization and Navigation using a Visual Memory

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

***Abstract*— A visual memory (VM) is a topological map in which a set of key images organized in form of a graph represents an environment. In this paper, a navigation strategy for humanoid robots addressing the problems of localization, visual path planning and path following based on a VM is proposed. Assuming that the VM is given, the main contributions of the paper are: 1) A novel pure vision-based localization method. 2) The introduction of the estimated rotation between key images in the path planning stage to benefit paths with enough visual information and with less effort of robot rotation. 3) The integration of the complete navigation strategy and its experimental evaluation with a Nao robot in an unstructured environment. The humanoid robot is modeled as a holonomic system and the strategy might be used in different scenarios like corridors, uncluttered or cluttered environments.**

I. INTRODUCTION

The robot navigation based on a visual memory (VM) is a method that mimics the human behavior of memorizing key scenes the first time that an environment is explored in order to facilitate a subsequent navigation in the same environment. A VM is a topological map that represents an environment by a set of *key images* [1], which are typically organized as a directed graph where each node is a key image and the edges provide information of the relation between nodes. Once this representation of the environment is known, the problems of robot localization, path planning and autonomous navigation can be solved. The navigation based on a VM can be considered as an extension of the typical visual servo-control task [2] that allows the robot to enlarge its workspace.

The visual servo-control is a local task that can be solved if the same visual scene is partially seen from a target and initial locations. This constraint can be avoided in a navigation scheme based on a VM, where a sequence of target images is used, such that the target and initial locations do not need to share information [1], [3]. Since the work in [4], where the view-sequenced route representation was introduced, some navigation strategies based on a VM have been suggested in the context of wheeled mobile robots [1], [5], [6]. For this kind of robots, this approach has been addressed with a position-based visual servoing (PBVS) scheme [5] or an image-based visual servoing (IBVS) scheme [1]. Also, direct feedback of a geometric constraint has been used in an IBVS [6].

[1]J. Delfin and G. Arechavaleta are with Robotics and Advanced Manufacturing Group, Centro de Investigación y de Estudios Avanzados del IPN, Saltillo, Coah. Mexico. {josafat.delfin,garechav}@cinvestav.mx

[2]H. M. Becerra is with Centro de Investigación en Matemáticas (CIMAT), C.P. 36023, Guanajuato, Gto., Mexico. hector.becerra@cimat.mx

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

Vision-based control of humanoid robots is an interesting and challenging problem due to the inherent complexity of the robotic system and the undesired sway motion introduced by the walking pattern [7]. The problem of visual servoing for humanoid robots has been addressed in works like [7]–[9]. These schemes exploit the reactive walking pattern generator (WPG) proposed in [10], which provides automatic generation of foot placements from desired values of linear and angular velocities of the robot's center of mass (CoM). In [8] and [9], a visual servoing scheme gives the reference of velocities for the CoM as the input to the WPG. Such schemes are decoupled since the visual controller is independent of the WPG. In [7], visual constraints are part of the WPG to define a coupled scheme.

Vision-based indoor navigation of humanoid robots for large displacements has also been addressed in the literature. A vision-guided locomotion scheme in structured indoor scenarios is proposed in [11], which drives the robot while avoiding obstacles. In [3], the view-sequenced route representation is used for humanoid navigation in a corridor. No initial localization is done and the controller is based on template correlation to decide next motion directions. Other humanoid navigation method based on pre-registered keyframes is suggested in [**?**], where known landmarks are used for localization. Another landmark-based navigation strategy that integrates motion planning through geometric primitives and visual servoing tasks is described in [12]. A trajectory tracking control based on vision fused with odometry is proposed in [13]. The trajectory is defined in the Cartesian space and the locomotion controller assumes the unicycle model. In [14], the visual navigation strategy deals with different corridor configurations exploiting vanishing points.

In this paper, we propose a navigation strategy based on a VM, addressing the problems of localization, visual path planning and path following. This is an extension of our previous work [15], where a control scheme for the visual path following stage has been proposed. Thus, assuming that the visual memory is known, i.e., the only available information is a set of key images, the main contributions of this paper are: 1) a vision-based localization method that relies on the homography decomposition; 2) we introduce a new way to define weights for the edges of the graph representing the VM. Besides of considering visual information (number of matched points), the estimated rotation between key images is also considered, such that the planning stage benefits paths with enough visual information between key images but with less effort of robot rotation; 3) the integration of the whole navigation scheme and its evaluation in an unstructured

environment. It is worth noting that the humanoid robot is considered as a holonomic system at the locomotion level, no motion constraints are imposed as in the unicycle model used in related works [12], [14]. Moreover, the proposed strategy can be used in less restrictive indoor environments. For instance: corridors, uncluttered or cluttered environments. Additionally, in the proposed localization method no extra information is needed, like odometry or sensor fusion, in comparison with methods like [13].

The paper is structured as follows. Sec. II gives an overview of the navigation strategy. Sec. III describes the structure of the VM. We present the localization strategy and path planning in Sec. IV. The control scheme is summarized in Sec. V. Experiments are presented in Sec. VI and conclusions are given in Sec. VII.

II. OVERVIEW OF THE NAVIGATION STRATEGY

The autonomous navigation framework presented in this work can be divided in four steps: 1) visual memory building; 2) robot localization; 3) discrete path planning; and 4) path following. A visual memory (VM) is represented by a directed graph $\mathbf{G} = \{\mathbf{I}, \mathcal{E}\}$ where each node is 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 that link the nodes. The generation of the VM is done offline, i.e., the set of key images is acquired on a supervised (human-guided) teaching phase.

The localization step consists of finding the image $\mathcal{I}_1^*$ of the VM that best fits the current image $\mathcal{I}$. We use the homography matrix to compute the measurements for finding the most similar key image in the graph. In the visual path planning stage, given an image of the VM as a target $\mathcal{I}_n^*$ and the image found by the localization stage, the planner finds the shortest visual path $\mathbf{I}^*$ linking those nodes (key images) of the graph.

The last step is the visual path following, which consists of autonomously executing the visual path found by the planner. The control guides the humanoid along the sequence of key images until the robot achieves the desired location. The main steps of the proposed framework are presented in Algorithm 1, where each function is detailed in the subsequent sections. We assume that the VM is given as an input of the algorithm. Thus, the memory building process is not addressed in this work.

III. STRUCTURE OF THE VISUAL MEMORY

As aforementioned, key images are selected during the teaching phase and they are the nodes of the graph $\mathbf{G}$. In this section, we complement the description of the structure of the graph by defining the edges between nodes.

An edge represents the cost of traveling from one node to any adjacent node, and it is set as a weight $\mathcal{E}_i$. We define the edges in terms of the number of matched interest points and the amount of rotation between neighboring key images as follows:

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

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

$\mathcal{I} =$ **captureCurrentImage**
$\mathcal{I}_1^* =$ **localizeRobot**($\mathcal{I}$, $\mathbf{I}$)
$\mathbf{I}^* =$ **getShortestPath**($\mathcal{I}_1^*, \mathcal{I}_n^*, \mathbf{G}$)
ROBOTNAVIGATION = **pathFollower**($\mathcal{I}$, $\mathbf{I}^*$)
return

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 (y-axis of the camera) between neighboring key images respectively, α and β are two 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. And the cost related to the rotation means that, the more rotation there are between nodes the higher the cost will be.

Thus, we need to estimate the relative θu_i^* between neighboring key images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$, with the associated reference frames $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$, respectively. An option to recover the whole relative pose (translation up to scale) between the camera frames $\mathcal{C}_i$ and $\mathcal{C}_{i+1}$ is by means of the homography matrix decomposition [16]. 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:

$$\mathbf{H}_i^* = \mathbf{R}_i^* + \frac{\mathbf{t}_i^*}{d_{i+1}^*}\mathbf{n}_{i+1}^* \tag{2}$$

where $\mathbf{R}_i^*$ and $\mathbf{t}_i^*$ are the rotation matrix and translation vector, d_{i+1}^* is the distance from a plane π to $\mathcal{C}_{i+1}$, and $\mathbf{n}_{i+1}^*$ 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 $\mathbf{R}_i^*$. Having the rotation matrix $\mathbf{R}_i^*$, it can be parametrized by the axis/angle $\theta\mathbf{u}_i^*$. Only the y-element of the vector $\theta\mathbf{u}_i^*$ is used to measure the rotation θu_i^* between neighboring key images since only that angular component is needed for the WPG as explained in [15].

An efficient algorithm to decompose a homography matrix is proposed in [17]. In this work, we rely on the homography decomposition to define the edges of the graph, to localize the robot and for visual control purposes. In particular, in order to deal with general 3D scenes, we use the algorithm proposed in [16], which allows us to estimate a homography associated to a virtual plane for non-planar scenes.

IV. VISUAL LOCALIZATION AND PLANNING

Once the visual memory (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 memorized key images. Next, given a target image, a path planning stage is needed to find a sequence of key images connecting the image resulting from the localization and the target image. In both stages, we

Algorithm 2: **localizeRobot** finds the most similar key image $\mathcal{I}_1^*$ in the graph 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 **for** $j \leftarrow 1$ ***to*** m **do**
2 $matches$ = **match**($\mathcal{I}$, $\mathcal{I}_j$)
3 **if** $matches > \mu$ **then**
4 $\mathbf{H}_j$ = **computeHomography**($matches$)
5 $(\mathbf{R}_j, \mathbf{t}_j)$ = **decomposeHomography**($\mathbf{H}_j$)
6 $\mathbf{t}_{\pm j}$ = **computeDirection**($\mathbf{t}_j$)
7 $\| \mathbf{t}_j \|$ = **computeDistance**($\mathbf{t}_j$)
8 $\mathbf{D}[j]$ = **saveImageData**($\mathbf{t}_{\pm j}$, $\| \mathbf{t}_j \|$)
9 **else**
10 NEXTIMAGE
11 $\mathbf{I}_+$ = **selectForwardImages**($\mathbf{D}$)
12 $\mathbf{I}_-$ = **selectBackwardImages**($\mathbf{D}$)
13 **if** SIZE($\mathbf{I}_+$) > 0 **then**
14 $\mathcal{I}_1^*$ = **selectMostSimilar**($\mathbf{I}_+$)
15 **else**
16 $\mathcal{I}_1^*$ = **selectMostSimilar**($\mathbf{I}_-$)
17 **return** $\mathcal{I}_1^*$
18 **Function** ***selectMostSimilar***($\mathbf{I}_\mu$)
19 $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ = **selectTwoCandidates**($\mathbf{I}_\mu$)
20 **if** $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ *belong to same branch* **then**
21 $matches$ = **match**($\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*, \mathcal{I}$)
22 $\{\mathbf{H}_1, \mathbf{H}_2\}$ = **parametersConstantPlane**($matches$)
23 $\{\| \mathbf{t}_1 \|, \| \mathbf{t}_2 \|\}$ = **computeDistance**($\{\mathbf{H}_1, \mathbf{H}_2\}$)
24 $\mathcal{I}_1^*$ = **selectTheClosest**($\{\| \mathbf{t}_1 \|, \| \mathbf{t}_2 \|\}$)
25 **else**
26 $\{\mathbf{I}_1, \mathbf{I}_2\}$ = **getShortestPath**($\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*, \mathcal{I}^*, \mathbf{G}$)
27 $\{\mathbf{d}_1, \mathbf{d}_2\}$ = **computeDistanceOfPath**($\{\mathbf{I}_1, \mathbf{I}_2\}$)
28 $\mathcal{I}_1^*$ = **getImageWithShortestPath**($\{\mathbf{d}_1, \mathbf{d}_2\}$)
29 **return** $\mathcal{I}_1^*$

take advantage of the homography decomposition described in Section III.

The visual localization process consists in finding the most similar key image of the memory in terms of the visual features by comparing it to the current image. We propose Algorithm 2 to localize the robot. First, **match** finds the matched features for every pair of images. Nevertheless, we have a minimum number of matches $\mu > 8$ to guarantee the computation of the homography and to have a subset of key images similar to $\mathcal{I}$. This is constrained due to the non-planar estimation of the homography [16].

With the matches, the next step is to compute the homography matrix and decompose it (lines 4-5). We classify the images for which the resulting direction is forward $\mathbf{I}_+$ or backward $\mathbf{I}_-$ with respect to the current image location. From the estimated translation vector $\mathbf{t}$, we assign a value in line 6 as follows:

$$t_\pm = \begin{cases} +1, & t_z > 0, \\ -1, & t_z < 0. \end{cases}$$

The relative distance between key images is computed by the norm $\| \mathbf{t} \|$ of the translation vector. Although the vector $\mathbf{t}$ is scaled, its norm gives a notion of distance. The

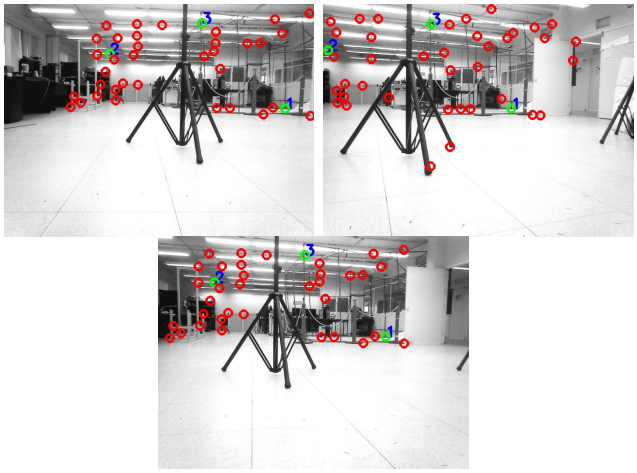


Fig. 1: **Constant virtual plane.** The green circles (marked with number) are selected and maintained as features to form the same virtual plane. **Top row.** The first candidate $\mathcal{I}_{\pm 1}^*$ and the second candidate $\mathcal{I}_{\pm 2}^*$. **Bottom row.** The current image $\mathcal{I}$.

direction and distance parameters are saved in a vector $\mathbf{D}$, from which the images are classified as forward $\mathbf{I}_+$ or backward $\mathbf{I}_-$ (lines 11-12). Using first the set $\mathbf{I}_+$, the function **selectMostSimilar** finds among them the most similar key image to the current one. If there are no forward images the algorithm selects the closest image among the backward images $\mathbf{I}_-$. We prefer to get the robot localization even if the most similar image is behind the current location.

The function **selectMostSimilar** (lines 18-29) first selects two candidate images $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$, the two images with the highest number of point matches. Using $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$, 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 $\| \mathbf{t} \|$. This process computes again $\mathbf{H}$ and $\mathbf{t}$ for the candidate images $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ with respect to the current image, but now: the virtual plane used for the homography estimation (according to [16]) remains constant (lines 22-23). Thus, the estimated distances are consistent. This process is illustrated in Fig. 1. Once the new distances $\{\| \mathbf{t}_1 \|, \| \mathbf{t}_2 \|\}$ are obtained, the closest candidate image $\mathcal{I}_1^*$ to the current one is selected (line 24), i.e., the image with smaller $\| \mathbf{t} \|$.

If the candidate images do not belong to the same branch, 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 26-28). This consideration is needed to discard a localization that might derive in a long path in the navigation stage. The process is as follows: using the graph $\mathbf{G}$, a path planner finds the shortest path from each one of the two candidates $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ to the target image $\mathcal{I}_n^*$. The distance of each path $\mathbf{I}_1, \mathbf{I}_2$ is computed and stored in a vector of distances $\{\mathbf{d}_1, \mathbf{d}_2\}$. The solution $\mathcal{I}_1^*$ of the localization will be the candidate image with the shortest path to the target image, i.e., the path with the minimum distance in terms of the edges defined in (1).

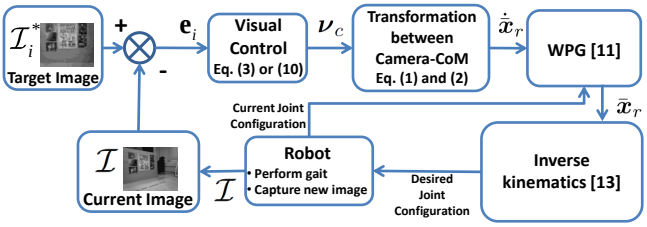


Fig. 2: Visual control scheme for humanoids.

The visual path $\mathbf{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \ldots, \mathcal{I}_{n-1}^*, \mathcal{I}_n^*\}$ represents the visual task that the robot has to accomplish. Given $\mathcal{I}_1^*$ and $\mathcal{I}_n^*$, the visual path is the set of key images that link the starting and the desired key images. The resulting visual path is the minimum length path in $\mathbf{G}$ that connects the nodes $\mathcal{I}_1^*$ and $\mathcal{I}_n^*$. The length of a path is the sum of the values of its edges, and the edges are defined according to (1). This process is done by the function **getShortestPath**.

V. VISUAL PATH FOLLOWING

Once the visual path is found, the vision-based walking controller is activated to follow autonomously the sequence of key images. We applied the scheme presented in our previous work [15]. Here we make a brief summary of that scheme.

The problem of visual path following can be treated as a set of n visual servoing subtasks where the visual error depends on the current image and the corresponding target image from the visual path $\mathbf{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \ldots, \mathcal{I}_{n-1}^*, \mathcal{I}_n^*\}$. In this work, we applied a PBVS, but the problem can be also solved by means of IBVS, as described in [15].

A. Visual control scheme for humanoids

We use a walking pattern generator (WPG) that considers automatic footstep placements and incorporates inequality linear constraints [10]. The motion coordination to generate the gait is computed by means of an efficient inverse kinematics method [18]. To generate the reference velocity $\dot{\tilde{\boldsymbol{x}}}_r$ for the WPG, let $\boldsymbol{\nu}_r$ be the velocity of the frame attached to the robot's CoM. We assume that a constant transformation ${}^r\boldsymbol{T}_c \in \mathbb{R}^{6\times6}$ exists between $\boldsymbol{\nu}_r$ and the velocity of the camera reference frame $\boldsymbol{\nu}_c$, thus:

$$\boldsymbol{\nu}_r = {}^r\boldsymbol{T}_c \boldsymbol{\nu}_c \in \mathbb{R}^6. \tag{3}$$

This constant transformation assumes that the robot's head is fixed w.r.t. the robot's body. Therefore, the input for the WPG can be expressed as:

$$\dot{\tilde{\boldsymbol{x}}}_r = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{\nu}_r \in \mathbb{R}^3. \tag{4}$$

Fig. 2 shows the whole process for the visual control tailored to humanoids. See [15] for more details.

Algorithm 3: pathFollower allows the robot to solve visual subtasks to perform the autonomous navigation.

Input: Visual path to follow $\mathbf{I}^* = \{\mathcal{I}_1^*, \mathcal{I}_2^*, \ldots, \mathcal{I}_{n-1}^*, \mathcal{I}_n^*\}$, current image $\mathcal{I}$
Output: Autonomous visual navigation

```
k ← 1
while k ≤ n do
    I*_k ← I*[k]
    matches = match( I*_k, I )
    H_k = computeHomography( matches )
    e_k = subtaskErrorComputation( H_k )
    ν_c = smoothTransitionControl( e_k )
    ROBOTGAIT = WPG( ν_c )
    ε_k = meanSquaredErrorComputation( I*_k, I )
    I = captureNewImage
    if ε_k < T_ε then
        k++
stopRobot
return
```

B. Position-based Visual Servoing

For each visual servoing task, the visual error that must be minimized can be defined as

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

where the visual features for a PBVS are given by [2]:

$$\mathbf{s} = (\mathbf{t}, \theta\mathbf{u}) \in \mathbb{R}^6 \tag{5}$$

being $\mathbf{t}$ the translation up to scale between the reference frames associated to the current camera pose $\mathcal{C}$ and the k^{th} key image frame $\mathcal{C}_k^*$, and $\theta\mathbf{u}$ the axis/angle parametrization of the rotation matrix $\mathbf{R}$ between those reference frames. We consider that the translation and rotation are expressed with respect to $\mathcal{C}_k^*$. Therefore $\mathbf{s}^* = 0$ and $\mathbf{e}_k = \mathbf{s}$. The vector $\mathbf{s}$ can be obtained by homography decomposition, as described at the end of Sec. III.

As detailed in [2], the translational and rotational velocities are decoupled and the velocity vector is given by:

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

If the camera pose is accurately estimated and the control gain $\lambda > 0$, the velocity vector (6) yields to an exponentially stable error dynamics of the form $\dot{\mathbf{e}} = -\lambda\mathbf{e}$.

The controller (6) can be introduced in the WPG according to Fig. 2, with the aim of driving the humanoid robot to the location associated to each key image $\mathcal{I}_k^*$ of the path $\mathbf{I}^*$.

C. Controller for smooth transition

In the visual path following problem there exists the issue that the error vector $\mathbf{s}$ in (5) suddenly increases when a new key image is given as target, which yields to discontinuous robot velocities [6]. In [15], we proposed a controller that achieves smooth transitions between subtasks. We introduced in (6) a smooth transition function $h(t)$ to penalize the large error at the beginning of each subtask. The proposed control law written in the form of the PBVS is given by:

$$\boldsymbol{\nu}_c = \begin{bmatrix} -\lambda h(t)\,\mathbf{t} \\ -\lambda h(t)\,\theta\mathbf{u} \end{bmatrix} \in \mathbb{R}^6. \tag{7}$$

The transition function $h(t)$ has the effect of a time-varying gain for the control law (6). An adequate proposal for a transition function is:

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

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 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 subtask.

Algorithm 3, shows the **pathFollower** function dedicated to this process. The point features $\mathbf{p}_j$ of the current image $\mathcal{I}$ are matched with the points $\mathbf{p}_j^*$ of the corresponding key image $\mathcal{I}_k^*$ along the robot motion. The matched points are used to compute the control law (7) from the homography decomposition. The switching to the next key image occurs when the mean squared error between the corresponding point features ε remains below a threshold T_ε over a finite number of iterations K, i.e.:

$$\varepsilon = \frac{1}{l}\sum_{j=1}^{l} \left\|\mathbf{p}_j - \mathbf{p}_j^*\right\| < T_\varepsilon, \tag{8}$$

where l is the number of corresponding point features in each subtask. The same steps are repeated for each key image in $\mathbf{I}^*$ until the target image $\mathcal{I}_n^*$ is reached.

VI. EXPERIMENTAL EVALUATION

We implemented the proposed navigation scheme on a Nao humanoid robot. The top camera mounted on the robot's head was used in the experiments. We made two experimental evaluations, one related to the localization algorithm and the boundaries where the robot can be initialized away from the visual paths and the other related to the whole navigation scheme, i.e., the localization, planning and visual path following.

For all the experimental evaluations the images were obtained with a resolution of 640 $\times$ 480 pixels. The image features were acquired as follows: first, a corner detector based on [19] was used, which is implemented in the function `goodFeaturesToTrack` of the OpenCV library. Then, we assigned a SURF descriptor [20] to each detected point. A robust matcher based on RANSAC matched all the points between the current image and the corresponding key image. The `HLM` function of ViSP library was used to compute the homography for planar and non-planar scenes [16]. All robot motions during the experiments were captured with an Optitrack System to obtain ground truth of the navigation scheme.

A. Localization

Here, we present the evaluation of the localization algorithm. We have made an experimental evaluation in order to know the boundaries around a key image where the localization algorithm can work effectively. We found that the working radius around a key image is approximately 0.7m.

1) Localization in one branch: First, we show the case when there is only one branch, i.e., there is only one visual path. Thus, if we refer to Algorithm 2, the condition in line 20 is always satisfied. So the localization process reduces to lines 20 to 24, where the algorithm selects the closest key image $\mathcal{I}_1^*$ to the current image $\mathcal{I}$.

We applied Algorithm 2 for each one of the three paths shown in Fig. 3. We classified the results in three categories, the case when closest key image was found ahead the robot, marked as "Forward", the case when the closest key image was found "Backward" the robot and the case when the localization was a "Failure", i.e., when none key image was found. Based on the results shown in Fig. 3, in all the cases the localization algorithm found a solution. In more than 80%, the closest image was found forward, this is preferred in order to perform a forward navigation from the beginning of the autonomous motion. Only 16% of the cases, the localization algorithm found a key image behind the robot, but still in these cases, a key image in the neighborhood of the robot was found.

2) Localization aided by a graph and planning: Now, we show the performance of the localization algorithm using a graph. In this case, the two candidate images $(\mathcal{I}_{\pm 1}^*, \mathcal{I}_{\pm 2}^*)$ might be in one branch or two different branches of the graph, thus the localization process is carried out by the lines 20 to 24 or 25 to 28 of Algorithm 2, respectively. The graph representing the VM is shown in Fig. 4, it has 90 key images and 94 edges. The green triangles represent the pose of the robot where each key image (node) was taken. The magenta triangles are the nodes that connect the branches of the graph. For the planning algorithm, we use as the function **getShortestPath**, Dijkstra's algorithm to obtain the minimum length path.

In Fig. 4, we present three cases of localization and planning. The first case is shown in Fig. 4(a). Here the robot starts near one branch (red triangle), however the candidate key images $\mathcal{I}_{\pm 1}^*$ and $\mathcal{I}_{\pm 2}^*$ (orange and gray triangles) are in the same branch and the localization algorithm selects the closest key image $\mathcal{I}_1^*$, in this case the orange mark. Once the robot is localized, **getShortestPath** finds the minimum length path $\mathbf{I}^*$ to the target image $\mathcal{I}_n^*$ (cyan triangle). The second case is shown in Figs. 4(b) and 4(c) . Here the robot starts between two branches of the graph (red triangle), so the candidate key images belong to different branches (orange and gray triangles). In this case, the localization algorithm is aided by the path planner to select the most similar key image accounting for the shortest path to the target image. As it can be seen in Fig. 4(b), the path to the target image (cyan triangle) is the shortest one if the algorithm selects the key image on the right (orange triangle), but if the

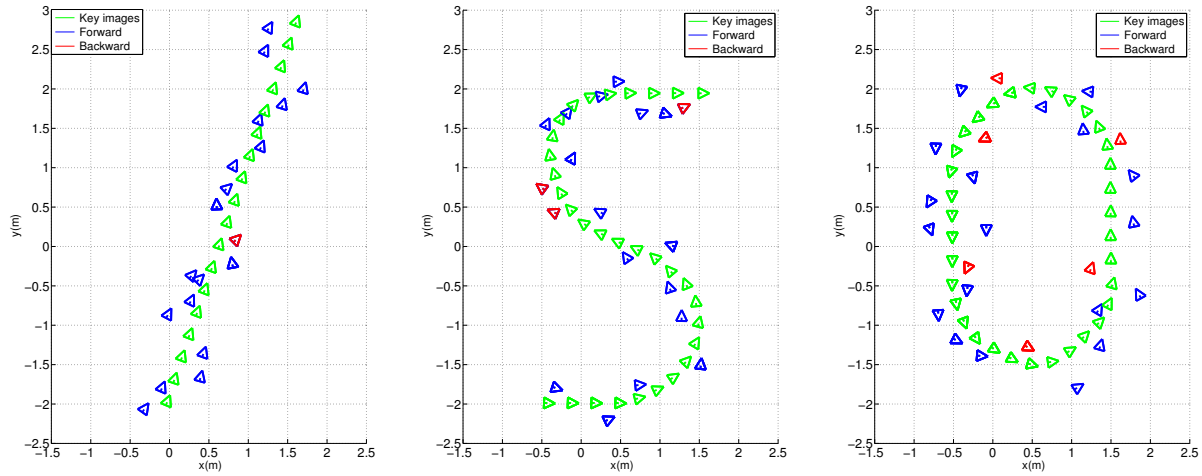
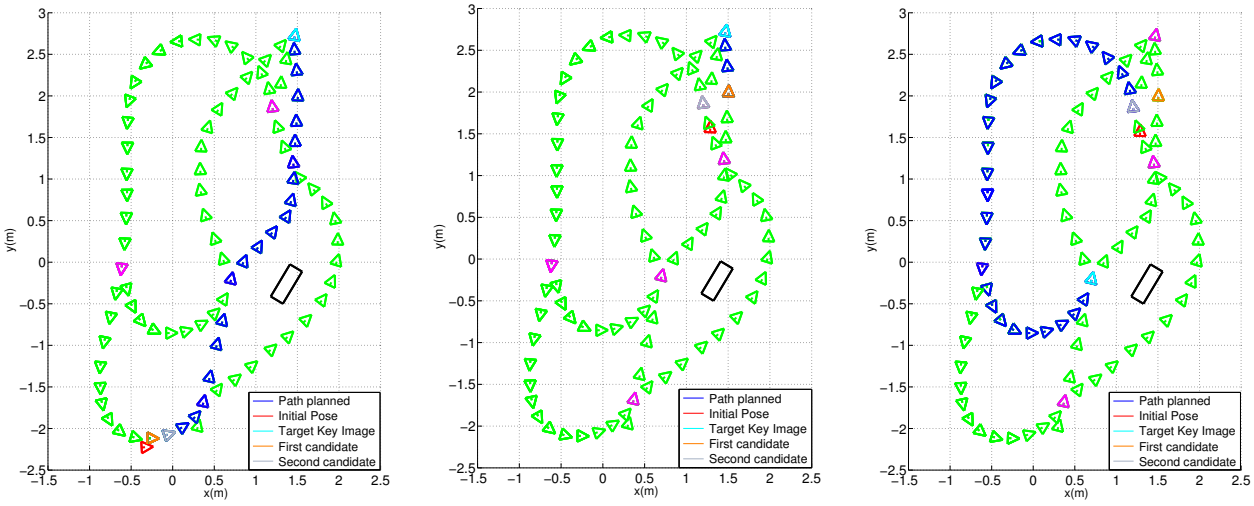


Fig. 3: **Results of localization in one branch.** We used three visual paths: a straight line, a S-like path and an elliptic path. The green triangles represent the robot's poses where key images were taken. The blue triangles represent the evaluation poses where the localization found forward images. The red triangles represent results of localization for backward images.



(a) (b) (c)

Fig. 4: **Evaluation of the localization in a graph. (a)** Case when two candidate images are in the same branch of the graph. **(b)** and **(c)** Cases when two candidate images are in different branches.

target image changes, as it is observed in the Fig. 4(c), the algorithm selects the key image on the left (gray triangle). This localization process avoids key images with longer paths associated to them.

B. Complete navigation scheme

Finally, we show the behavior of the robot for the whole navigation scheme, i.e., once the robot was localized and the path was found, the navigation is carried out by means of the visual path following scheme described in Sec. V. For this experiment, we use the same image features described at the beginning of this section but also we use 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 tracker with the Nao robot because of the jerky camera movements generated by the robot's gait. Thus, the tracker was experimentally tuned to deal with this unavoidable behavior.

Fig. 5 shows the performance of the navigation scheme for the humanoid robot Nao. The red triangle depicts the initial pose of the robot, the gray triangle is the most similar key image that Algorithm 2 returned, the target key image is the cyan triangle and the orange triangle is the final pose of the robot during the experiment. The visual path planned was composed by 23 key images over a total distance close to 6m. The localization takes approximately 7 seconds to compare all key images in the graph with the current image.

Concerning the control parameters we used $\lambda = 0.065$ for translation and $\lambda = 0.3$ for rotation. They are different because in bipedal locomotion, it might be easier for the

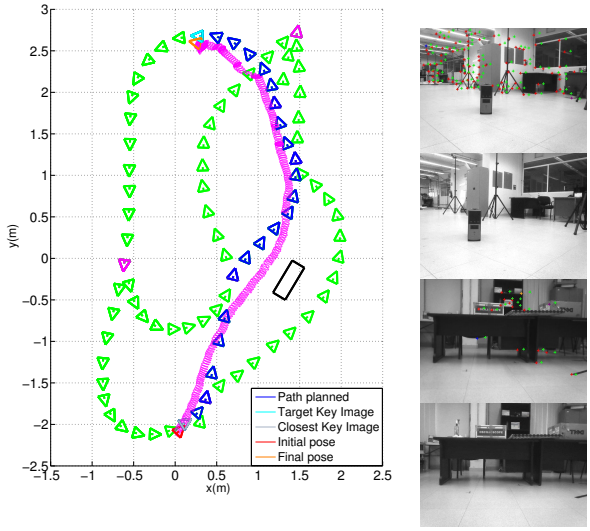


Fig. 5: **Performance of the autonomous robot navigation.** The magenta line shows the motion of the robot during the navigation. As it can be seen, the robot gets close to the target key image (cyan triangle) with an error of less than 10cm. The images on the right from top to bottom are: the current image $\mathcal{I}$ at the initial pose (red triangle), the closest key image $\mathcal{I}_1^*$ (gray triangle), the final pose (orange triangle) and the target key image $\mathcal{I}_n^*$ (cyan triangle).

robot to move laterally compared to rotate. The duration of the transition function $h(t)$ was set experimentally to 40 iterations. Also, for a good compromise between mitigation of discontinuities of walking velocities and continuity of motion during navigation, we assigned experimentally $h_0 = 0.2$. The threshold T_ε was set to 18 pixels for the whole path and $K = 100$ iterations.

We evaluated the controller presented in Sec. V-B, but it is important to mention that also an IBVS can be used for the navigation as presented in our previous work [15]. Overall, the proposed scheme solves a global navigation task by driving the robot to a vicinity of the location associated to the target key image, which is the main goal rather than to follow the path with high accuracy.

VII. CONCLUSIONS

In this paper, we have proposed a navigation scheme for humanoid robots using the appearance-based environment representation called visual memory. Assuming that a set of key images which defines a visual memory is known, the navigation scheme consists of three stages: localization, planning and execution. Each stage only relies on visual information based on the homography matrix decomposition for planar and non planar scenes. Besides the visual information considered as matched point features, we introduced an estimated rotation between key images in the path planning stage to benefit visual paths with less rotations. The stages have been integrated and evaluated in a real-world setting with a Nao humanoid robot. We have shown experimentally that our navigation scheme works for humanoid robots in unstructured indoor environments.

For future work, we consider extending our scheme for long distance navigation and handling a hierarchical task scheme to deal with different types of obstacle avoidance.

REFERENCES

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

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

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

[4] Y. Matsumoto, M. Inaba, and H. Inoue. Visual navigation using view-sequenced route representation. In *IEEE Int. Conf. on Robotics and Automation*, pages 83–88, 1996.

[5] E. Royer, M. Lhuillier, M. Dhome, and J. M. Lavest. Monocular vision for mobile robot localization and autonomous navigation. *Int. Journal of Computer Vision*, 74(3):237–260, 2007.

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

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

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

[9] J. Delfin, H. M. Becerra, and G. Arechavaleta. Visual servo walking control for humanoids with finite-time convergence and smooth robot velocities. *Int. Journal of Control*, 34(4–5):402–419, 2014.

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

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

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

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

[14] A. Paolillo, A. Faragasso, G. Oriolo, and M. Vendittelli. Vision-based maze navigation for humanoid robots. *Autonomous Robots*, pages 1–17, 2016.

[15] 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 Robots (Humanoids)*, pages 354–359, 2014.

[16] E. Malis, F. Chaumette, and 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):79–97, 2000.

[17] B. Triggs. Autocalibration from planar scenes. In H. Burkhardt and B. Neumann, editors, *Computer Vision - ECCV'98*, volume 1406 of *LNCS*, pages 89–105. Springer Berlin Heidelberg, 1998.

[18] O. Kanoun. Real-time prioritized kinematic control under inequality constraints for redundant manipulators. In *Robotics: Science and Systems VII*, Los Angeles, CA, USA, June 2011.

[19] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[20] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *European Conf. on Computer Vision*, pages 404–417, 2006.