Gaze Direction Determination of Opponents and Teammates in Robot Soccer*

Patricio Loncomilla[1] and Javier Ruiz-del-Solar[1,2]

[1] Department of Electrical Engineering, Universidad de Chile
[2] Center for Web Research, Department of Computer Science, Universidad de Chile
{ploncomi, jruizd}@ing.uchile.cl

Abstract. Gaze direction determination of opponents and teammates is a very important ability for any soccer player, human or robot. However, this ability is still not developed in any of the RoboCup soccer leagues. We aim at reverting this situation by proposing a gaze direction determination system for robot soccer; the system is designed primarily for the four-legged league, but it could be extended to other leagues. This system is based on a robot-head pose detection system, consisting on two main processing stages: (i) computation of scale-invariant local descriptors of the observed scene, and (ii) matching of these descriptors against descriptors of robot-head prototypes already stored in a model database. After the robot-head pose is detected, the robot gaze direction is determined using a head model of the observed robot, and the current 3D position of the observing robot camera. Experimental results of the proposed approach are presented.

1 Introduction

Among many other capabilities, good soccer players should have the ability for anticipating the actions of opponents, and sometimes of teammates, by just observing the other players attitude and pose. As in other similar situations, the human most employed mechanism for solving this task is gaze direction determination, or the determination of the place where the opponent or teammate player under analysis is looking. For instance, by using this mechanism an attacker player can know if an opponent is observing him, and then planning his next actions for avoiding that the opponent attack him or obstruct his trajectory. In another typical situation a soccer player can know where the ball is, by looking at the same position where an opponent is looking (in case the opponent knows the ball position). In a third situation a soccer player can send the ball, i.e. perform a pass, to a position where a teammate is looking at. Furthermore, when kicking the ball, first-class soccer players can mislead opponents by looking at a different place than the place where they are sending the ball. Some examples of these typical situations are shown in figure 1.

* This research was funded by Millenium Nucleus Center for Web Research, Grant P04-067-F, Chile.

A. Bredenfeld et al. (Eds.): RoboCup 2005, LNAI 4020, pp. 230 – 242, 2006.

On hand of the described situations, it can be affirmed that gaze direction determination of opponents and teammates is a very important ability for any soccer player, robot or human. However, this ability is still not developed in any of the RoboCup soccer leagues. We aim at reverting this situation by proposing a gaze direction determination system for robot soccer. This system is designed primarily for the four-legged league, but it could be extended to other leagues. Moreover, the same gaze determination methodology can be used for enhancing cooperative and competitive skills in situations where the robots interacting abilities are important.

In the here-proposed approach, gaze direction determination is based on a robot-head pose detection system. This detection system employs two main processing stages. In the first stage, scale-invariant local descriptors of the observed scene are computed. Then, in the second stage these descriptors are matched against descriptors of robot-head prototypes already stored in a model database. After the robot-head pose is recognized, the robot gaze direction is determined using a head model of the observed robot, and the current 3D position of the observing robot camera. In the here-employed robots (Sony AIBO) the relation between head and camera pose is fixed, therefore additional camera pose determination is not required.

The local descriptors computation and matching are based on [1], but many important parts of the method have been improved for fitting it to the robot-head detection problem and for achieving high detection accuracy.

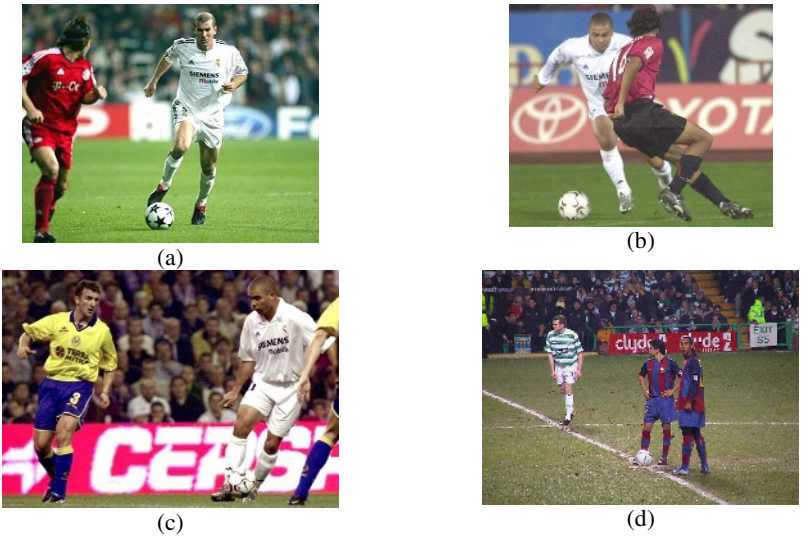

Fig. 1. Some examples of real soccer situations where the gaze direction determination plays an important role. (a) and (b) An attacker player knows if an opponent is observing him and at which distance. (c) A defender knows where the ball is, by looking at the same place where the attacker is looking. (d) Soccer players can mislead opponents by looking at a different place than the place where they are sending the ball.

2 Related Work

Human gaze direction (i.e. line of gaze) determination has been the subject of a great number of studies (for example [3][4][8][9]), with applications in very different fields such as medical research for oculography determination, car drivers behavior characterization, human-robot and human-computer interaction, including computer interfaces for handicapped people. However, to our knowledge there are no studies on determining the gaze direction in robots. We believe this is a problem that needs to be solved for enhancing and enriching cooperative and competitive tasks in which the robots interacting capabilities are important (i.e. robot soccer). Already developed methodologies employed for human gaze direction determination are not applicable for robots. They are based on anthropometric models of the human head and eyes, or they employ face or iris detection algorithms, or even special lighting (infrared lights). Therefore, new methodologies need to be employed for the robot case. Some alternatives could be the construction of explicit 3D robot-head models, the development of robot-face detection algorithms or the use of scale-invariant local descriptors for performing the detection. Taking into account the impressive development of object recognition algorithms based on scale-invariant descriptors in the last years ([1][6][7]), and the fact that head and face variability in robots is much smaller than in humans, we believe that for the moment, they are the best methodology for solving this problem. Most successful proposed systems employ either the Harris detector [5] or SIFT (Scale Invariant Feature Transform) features [1] as building blocks. In this work we employ SIFT features because of their higher robustness and stability. However, due to the physical characteristics of some robots models as the AIBO ERS7 (rounded head shape and head surface producing a high amount of highlights), it is very difficult to obtain reliable SIFTs on them. For this reason, we improve the traditional SIFTs computation and matching algorithms.

3 Proposed Robot Gaze Direction Determination System

3.1 Scale-Invariant Local Descriptors Computation

Detection of scale-space extrema. A difference-of-Gaussian (DoG) function is employed for identifying potential interest points that are invariant to scale and orientation. These keypoints are searched over all scales and image locations using a scale-space transformation. It can be proved that by using the DoG over the scale-space, image locations that are invariant to scales can be found, and that these features are more stable than other computed using the gradient, Hessian or Harris corner function [1]. The scale-space of an image is defined as a function, $L(x,y,\sigma)$, which corresponds to the convolution of the image with a Gaussian of scale σ. The DoG function between two nearby scales separated by a constant multiplicative factor k can be computed as:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

The local extrema (maxima and minima) of $L(x,y,\sigma)$ are detected by comparing each sample with its 26 neighbors in the scale-space (8 in the same scale, 9 in the scale above and 9 in the scale below).

Accurate keypoint localization. First, local extrema to sub-pixel / sub-scale accuracy are found by fitting a 3D quadratic to the scale-space local sample point. The quadratic function is computed using a second order Taylor expansion having the origin at the sample point [2]:

$$D(\mathbf{x}) = D(\mathbf{0}) + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{1}$$

where **x** is the offset from the sample point. Then, by taking the derivate with respect to **x** and setting it to zero, the location of the extrema of this function is given by:

$$\hat{\mathbf{x}} = -H^{-1}\nabla D(\mathbf{0}) \tag{2}$$

In [1][2] the Hessian and gradient are approximated by using differences of neighbor samples points. The problem with this coarse approximation is that just 3 samples are available in each dimension for computing the Hessian and gradient using pixel differences, which produces a non-accurate result. We improve this computation by using a real 3D quadratic approximation of the scale-space, instead of discrete pixel differences. Our 3D quadratic approximation function is given by:

$$\tilde{D}(x,y,\sigma) = a_1x^2 + a_2y^2 + a_3\sigma^2 + a_4xy + a_5x\sigma + a_6y\sigma + a_7x + a_8y + a_9\sigma + a_{10}$$

Using the 27 samples contained in the 3x3x3 cube under analysis, the unknowns (a_i) can be found. Using vector notation, this linear system will be given by:

$$\begin{bmatrix} x_1^2 & y_1^2 & \sigma_1^2 & x_1y_1 & x_1\sigma_1 & y_1\sigma_1 & x_1 & y_1 & \sigma_1 & 1 \\ x_2^2 & y_2^2 & \sigma_2^2 & x_2y_2 & x_2\sigma_2 & y_2\sigma_2 & x_2 & y_2 & \sigma_2 & 1 \\ & & & & \cdots & & & & & \\ x_{27}^2 & y_{27}^2 & \sigma_{27}^2 & x_{27}y_{27} & x_{27}\sigma_{27} & y_{27}\sigma_{27} & x_{27} & y_{27} & \sigma_{27} & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_{10} \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ \cdots \\ D_{27} \end{bmatrix}$$

where D_i corresponds to the sample point value (intensity) i. We can write this linear system as $\mathbf{Ba} = \mathbf{d}$. The least-squares solution for the parameters **a** is given by:

$$\mathbf{a} = \left(\mathbf{B}^\mathbf{T}\mathbf{B}\right)^{-1}\mathbf{B}^\mathbf{T}\mathbf{d}$$

It should be stressed that the matrix $\left(\mathbf{B}^\mathbf{T}\mathbf{B}\right)^{-1}\mathbf{B}^\mathbf{T}$ needs to be computed once for the whole image, and that it can be eventually pre-computed. Now, the accurate location of the extrema can be computed using (2), with the following Hessian and gradient expression:

$$\mathbf{H} = \begin{bmatrix} 2a_1 & a_4 & a_5 \\ a_4 & 2a_2 & a_6 \\ a_5 & a_6 & 2a_3 \end{bmatrix};\ \nabla\tilde{D}(0) = \begin{bmatrix} a_7 \\ a_8 \\ a_9 \end{bmatrix} \tag{3}$$

Second, local extrema with a contrast lower (noise) than a given threshold Th_{contr}, are discarded ($\left|\tilde{D}(\hat{\mathbf{x}})\right| < Th_{contr}$). Third, extrema corresponding to edges are discarded using curvature analysis. A peak that corresponds to an edge will have a large principal curvature across the edge but a small one in the perpendicular direction. The curvature can be computed from the 2x2 submatrix $\mathbf{H}_{xy}$ that considers only the x and y components of the Hessian. Taking into account that we are interested on the ratio between the eigenvalues, we will discard extrema in which the ratio of principal curves is above a threshold r, or equivalently local extrema that fulfill the following condition (see [5] for a deeper explanation):

$$\frac{\mathrm{Tr}(\mathbf{H}_{xy})^2}{\mathrm{Det}(\mathbf{H}_{xy})} > \frac{(r+1)^2}{r}$$

In [1] $\mathbf{H}_{xy}$ is computed be taking differences of neighbor sample points. As already mentioned, this approximation produces a non-accurate result. We improved this situation by computing $\mathbf{H}_{xy}$ from (3).

Orientation assignment. By assigning a coherent orientation to each keypoint, the keypoint descriptor can be represented relative to this orientation and hence achieve invariance against rotations. The scale of the keypoint is employed for selecting the smoothed image L(x,y) with the closest scale, and then the gradient magnitude and orientation are computed as:

$$\mathrm{m}(x,y) = \sqrt{(L(x+1,y)-L(x-1,y))^2 + (L(x,y+1)-L(x,y-1))^2}$$
$$\theta(x,y) = \tan^{-1}((L(x,y+1)-L(x,y-1))/(L(x+1,y)-L(x-1,y)))$$

As in [1], an orientation histogram is computed from the gradient orientations at sample points around the keypoint (*b1* bins are employed). A circular Gaussian window whose size depends of the scale of the keypoints is employed for weighting the samples. Samples are also weighted by its gradient magnitude. Then, peaks in the orientation histogram are detected: the highest peak and peaks with amplitudes within 80% of the highest peak. Orientations corresponding to each detected peak are employed for creating a keypoint with this orientation. Hence, multiple keypoints with the same location and scale but different orientation can be created (empirically, about 85% of keypoints have just one orientation).

Keypoint descriptor computation. For each obtained keypoint, a descriptor or feature vector that considers the gradient values around the keypoint is computed. The obtained descriptors are invariant against some levels of change in 3D viewpoint and illumination. The keypoints and their associated descriptors are knows as SIFT (Scale Invariant Feature Transform) features or just SIFTs.

First, in the keypoint scale the gradient magnitude and orientation are computed around the keypoint position (usually a neighborhood of 8x8 or 16x16 pixels is considered). Then, a Gaussian window weights the gradient magnitudes, and the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. Second, the obtained gradient values are accumulated in orientation histograms summarizing the contents of 4x4 subregions (*b2* bins are employed). Thus, a descriptor vector is built, where each vector component is given

by an orientation histogram. Depending on the neighborhood size, 2x2 or 4x4 vectors are obtained. Third, illumination effects are reduced by normalizing the descriptor's vector to unit length. Abrupt brightness changes are controlled by limiting the intensity value of each component of the normalized vector. Finally, descriptors vectors are re-normalized to unit length.

3.2 Matching of Local Descriptors and Robot-Head Prototypes Descriptors

Basically, the robot-head pose is determined by matching the image descriptors with descriptors corresponding to robot-head prototype images already stored in a database. The employed prototypes correspond to different views of a robot head, in our case the head of an AIBO ERS7 robot. Due to we are interested in recognized the robot identity (number), prototypes for each of the four players are stored in the database. In figure 2 are displayed the 16 prototype heads corresponding to one of the robots. The pictures were taken every 22.5°. The whole matching process here-proposed considers nine processing stages. In the first stage, the image keypoint descriptors are individually matched against prototype descriptors. In the second stage this matching information is employed for obtaining a coarse prediction of the object (robot-head) pose. In the third stage possible affine transformations between a prototype and the located object are determined. In the later six stages these affine transformations are verified, and some of them discarded or merged. Finally, if the object is present in the image just one affine transformation should remain. This transformation determines the object pose. It is worth to mention than in the original work of Lowe [1], only the first four stages here-described were considered. We included five additional verification stages that improve the detection of robot heads. This is very important because due to the physical characteristics of the AIBO ERS7 heads (rounded head shape, head surface producing a high amount of highlights, etc.), it is very difficult to obtain reliable SIFTs on them.

Individual Keypoint Descriptors Matching. The best candidate match for each image keypoint is found by computing its Euclidian distance with all keypoints stored in the database. It should be remembered that each prototype includes several keypoint descriptors. Considering that not all keypoints are always detected (changes in illumination, pose, noise, etc.) and that some keypoints arise from the image background and from other objects, false matches should be eliminated. A first alternative is to impose a minimal value to a match to be considered correct. This approach has proved to be not robust enough. A second alternative consists on comparing the distance to the closest neighbor to that of the second-closest neighbor. If this ratio is greater than a given threshold, it means than this image keypoint descriptor is not discriminative enough, and therefore discarded. In [1] the closest neighbor and second-closest neighbor should come from a different object model (prototype). In the current case this is not a good idea, since we have multiple views of the same object (the robot head). Therefore, we impose the conditions than the second-closest neighbor should come from the same prototype than the closest neighbor. The image under analysis and the prototype images generate a lot of keypoints, hence having an efficient algorithm for computing the keypoint descriptors distance is a key issue. This nearest neighbor indexing is implemented using the Best-Bin-First algorithm [10], which employs a k-d tree data structure.

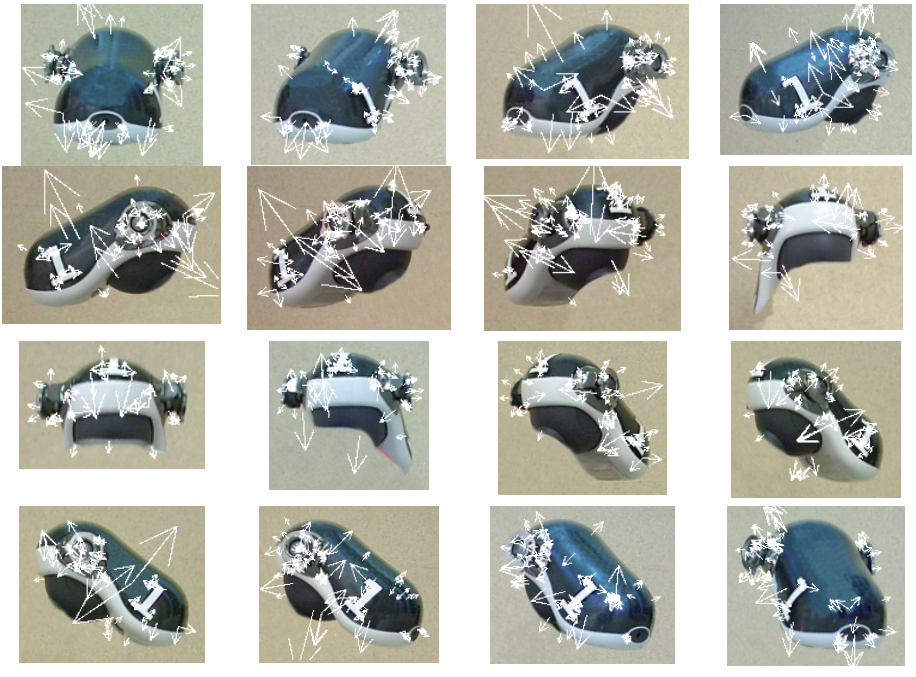

Fig. 2. AIBO ERS7 robot-head prototypes with their SIFTs. Pictures taken every 22.5°.

Object Pose Prediction. In the pose space a Hough transform is employed for obtaining a coarse prediction of the object (robot-head) pose, by using each matched keypoint for voting for all object pose that are consistent with the keypoint. A candidate object pose is obtained if at least 3 entries are found in a Hough bin. Usually, several possible object pose are found. The prediction is coarse because the similarity function implied by the four parameters (2D location, orientation and scale) is only an approximation of the 6 degree-of-freedom of a 3D object. Moreover, the similarity function cannot account for non-rigid deformations.

Finding Affine Transformations. In this stage already obtained object pose are subject to geometric verification. A least-squares procedure is employed for finding an affine transformation that correctly account for each obtained pose. An affine transformation of a prototype keypoint (x,y) to an image keypoint (u,v) is given by:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

where the m_i represent the rotation, scale and stretch parameters, and t_x and t_y the translation parameters. The parameters can be found if three or more matched keypoints are available. Using vector notation, this linear system will be given by:

$$\begin{pmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \dots & & & \\ & & \dots & & & \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{pmatrix} = \begin{pmatrix} u \\ v \\ \dots \\ \dots \end{pmatrix}$$

We can write this linear system as $\mathbf{Cp} = \mathbf{u}$. Finally, the least-squares solution for the parameters $\mathbf{p}$ is given by:

$$\mathbf{p} = \left(\mathbf{C}^{\mathrm{T}}\mathbf{C}\right)^{-1}\mathbf{C}^{\mathrm{T}}\mathbf{u}.$$

Affine Transformations Verification using a Probabilistic Model. The obtained model hypothesis, i.e. affine transformations, is subject to verification using a probabilistic model (see detailed description in [11]).

Affine Transformations Verification based on Geometrical Distortion. A certain affine transformation shouldn't deform very much an object when mapping it. Given that we have just a hypothesis of the object pose, it is not easy to determine the object distortion. However, we do have the mapping function, i.e. the affine transformation. Therefore, we can verify if the mapping function produce distortion or not using a known, regular and simple object, such as a square. The affine transformation of a square should produce a rotated parallelogram. If the affine transformation does not produce a large distortion, the conditions that the transformed object should fulfill are (see notation in fig. 3):

$$\max\left\{\frac{d(AB)/d(A'B')}{d(BC)/d(B'C')}, \frac{d(BC)/d(B'C')}{d(AB)/d(A'B')}\right\} < th_{prop} \;;\; \alpha = \sin^{-1}\left|\frac{\det(\overrightarrow{A'B'} \quad \overrightarrow{B'C'})}{d(\mathrm{A'B'}) \times d(\mathrm{B'C'})}\right| > th_{\alpha}$$

$\overrightarrow{A'B'}$ is a vector from *A'* to *B'*, $\det\left(\overrightarrow{A'B'} \quad \overrightarrow{B'C'}\right)$ computes the parallelogram area.

Affine Transformations Verification based on Spatial Correlation. Affine transformations producing low lineal correlation, r_s, between the spatial coordinates of the matched SIFTs in the image and in the prototype are discarded:

$$r_s = \min\left(\max(r_{xx}, r_{xy}), \max(r_{yx}, r_{yy})\right) < th_{rs}$$

r_{xx} and r_{yy} correspond to the correlation in the x and y directions of the *N* matched SIFTs, while r_{xy}=r_{yx} corresponds to the cross correlation between both directions. r_{xx} and r_{xy} are calculated as (r_{yy} and r_{yx} are computed in a similar way):

$$r_{xx} = \left| \frac{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)\left(x'_i - \overline{x'}\right)}{\sqrt{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2 \sum_{i=1}^{N}\left(x'_i - \overline{x'}\right)^2}} \right| ; \; r_{xy} = \left| \frac{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)\left(y'_i - \overline{y'}\right)}{\sqrt{\sum_{i=1}^{N}\left(x_i - \overline{x}\right)^2 \sum_{i=1}^{N}\left(y'_i - \overline{y'}\right)^2}} \right|$$

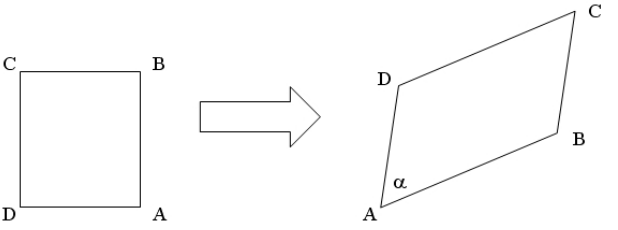

Fig. 3. Affine mapping of a square

Affine Transformations Verification based on Graphical Correlation. Affine transformations producing low graphical correlation, r_g, between the robot-head prototype image and the candidate robot-head subimage can be discarded:

$$r_g = \frac{\sum_{u=0}^{U}\sum_{v=0}^{V}\left(I(u,v)-\bar{I}\right)\left(I'\left(x_{TR}(u,v),y_{TR}(u,v)\right)-\bar{I'}\right)}{\sqrt{\sum_{u=0}^{U}\sum_{v=0}^{V}\left(I(u,v)-\bar{I}\right)^2\sum_{u=0}^{U}\sum_{v=0}^{V}\left(I'\left(x_{TR}(u,v),y_{TR}(u,v)\right)-\bar{I'}\right)^2}} < th_{rg}$$

The affine transformation is given by $\{x=x_{TR}(u,v),\ y=y_{TR}(u,v)\}$. $I(u,v)$ and $I'(x,y)$ correspond to the head prototype image and the candidate robot-head subimage, respectively. $\bar{I}$ and $\bar{I'}$ are the corresponding pixel mean values.

Affine Transformations Verification based on the Object Rotation. In some real-world situations, real objects can have restrictions in the rotation (respect to the body plane) they can suffer. For example the probability that a real soccer robot is rotated in 180° (inverted) is very low. For a certain affine transformation, the rotation of a detected object (candidate robot-head) with respect to a certain prototype can be determined using the SIFTs keypoint orientation information. Thus, the object rotation, *rot*, is computed as the mean value of the differences between the orientation of each matched SIFTs keypoint in the prototype and the corresponding matched SIFTs keypoint in the image. Transformations producing large *rot* values can be discarded ($rot > th_{rot}$).

Affine Transformations Merging based on Geometrical Overlapping. Sometimes more than one correct affine transformation corresponding to the same object can be obtained. There are many reasons for that, small changes in the object view respect to the prototypes views, transformations obtained when matching parts of the object as well as the whole object, etc. When these multiple, overlapping transformations are detected, they should be merged. As in the case when we verify the geometrical distortion produce by a transformation, we perform a test consisting in the mapping of a square by the two candidate affine transformations to be joined. The criterion for joining them is the overlap, *over*, of the two obtained parallelograms (see notation in fig. 3):

$$over = 1 - \frac{dist(A'_1A'_2)+dist(B'_1B'_2)+dist(C'_1C'_2)+dist(D'_1D'_2)}{perimeter(A'_1B'_1C'_1D'_1)+perimeter(A'_2B'_2C'_2D'_2)} > th_{over}$$

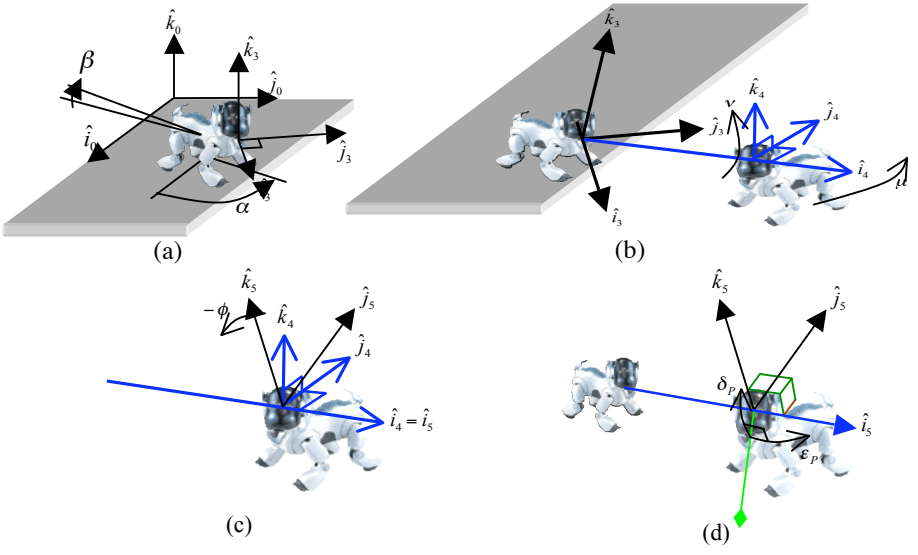

Fig. 4. Defined reference systems (RFs) (see explanation in main text). (a) RFs "0" and "3". (b) RFs "3" and "4", rotation angles μ and ν. (c) RFs "4" and "5", and affine rotation angle ϕ. (d) Observed robot in RF "5", line of gaze in green.

It should be also verified if the difference between the rotations produced for each transform is not very large. Therefore, two transforms will be joined if:

$$|rot_1 - rot_2| < th_{diff\_rot}.$$

3.3 Gaze Determination

The line of gaze of the observed robot, in global coordinates, can be computed using the following information: (i) observing robot pose in global coordinates, (ii) prototype view angle, and (iii) distance and rotation angle of the observed robot. The observing robot pose can be estimated using the self-localization system (any mobile robot control software has a similar system). The prototype view angle is fixed and known, it was defined when the model database was built. Finally, the distance and rotation angle of the observed robot can be determined from the affine transformation.

For performing the computations we define the following coordinate systems: $\{\hat{i}_0\ \hat{j}_0\ \hat{k}_0\}$, the global reference system (RF), $\{\hat{i}_3\ \hat{j}_3\ \hat{k}_3\}$, a RF fixed to the observing robot's camera (between system "0" and "3" there are 3 coordinate transformations), and $\{\hat{i}_5\ \hat{j}_5\ \hat{k}_5\}$, a RF located at the observed robot's head (between system "3" and "5" there are 2 coordinate transformations). The considered angles and distances are defined in table1.

In the RF 5, two points will define the line of gaze: the camera's position of the observed robot, and the intersection of the gaze straight line (of parameter λ) with the floor. This straight line will be given in system "5" coordinates by:

$$x_5(\lambda) = -\lambda \cos\delta_P \cos\varepsilon_P \ ; \ y_5(\lambda) = -\lambda \cos\delta_P \sin\varepsilon_P \ ; \ z_5(\lambda) = \lambda \sin\delta_P$$

Table 1. Angles and distances definitions

	Definition	**Source**
α	Rotation angle of robot's body with respect to axis $\hat{i}_0$ with rotation axis $\hat{k}_0$	Self-localization
β	Elevation angle of robot's body with respect to plane $\{\hat{i}_0 \ \hat{j}_0\}$	Accelerometer
γ	Tilt 1 angle (body–neck angle)	Robot joints
δ	Tilt 2 angle (neck–head angle)	Robot joints
ε	Head's pan angle (neck–head angle)	Robot joints
δ_P	Prototype head tilt angle in the matched image	Prototype angle
ε_P	Prototype head pan angle in the matched image	Prototype angle
l_1	Neck's longitude (distance between the two rotation centers of the neck)	Robot geometry
l_2	Head's longitude (distance between neck-head rotation center and the camera)	Robot geometry
P	3D position of the observing robot body–neck rotation center measured from the global reference system	Self-localization
C	3D position of the observing robot's camera measured from the global reference system. This point corresponds to the origin of reference system 3.	Self-localization
R	3D position of the observed robot head measured from the observing robot camera. Measured in reference system 3.	To be computed
μ	Horizontal angle of the straight line that joints the two robot-heads, measured from the "3" reference system (see figure 4 (b)).	To be computed
ν	Vertical angle of the straight line that joints the two robot-heads, measured from the "3" reference system (see figure 4 (b)).	To be computed
ϕ	Affine transformation associated rotation. Computed using the mean of the SIFT angle differences in all the keypoints matches used to compute the transformation (named previously *rot*).	Robot head-pose determination system

These equations can be translated to global coordinates (x_0, y_0, z_0) using coordinate transformations. The intersection with the body will correspond, in global coordinates, to: $z_0(\lambda) = 0$. Then, going from RF "5" to RF "0" is given by:

$$\begin{pmatrix} x_0 & y_0 & z_0 & 1 \end{pmatrix}^T = M_{10} M_{21} M_{32} M_{43} M_{54} \begin{pmatrix} x_5 & y_5 & z_5 & 1 \end{pmatrix}^T$$

with:

$$M_{10} = \begin{pmatrix} \cos\alpha\cos\beta & -\sin\alpha & \cos\alpha\sin\beta & P_x \\ \sin\alpha\cos\beta & \cos\alpha & \sin\alpha\sin\beta & P_y \\ -\sin\beta & 0 & \cos\beta & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix}; \quad M_{21} = \begin{pmatrix} -\sin\gamma & 0 & -\cos\gamma & -l_1\sin\gamma \\ 0 & 1 & 0 & 0 \\ \cos\gamma & 0 & -\sin\gamma & l_1\cos\gamma \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{32} = \begin{pmatrix} \sin\delta & 0 & \cos\delta & l_2\sin\delta \\ \cos\delta\sin\varepsilon & \cos\varepsilon & -\sin\delta\sin\varepsilon & l_2\cos\delta\sin\varepsilon \\ -\cos\delta\cos\varepsilon & \sin\varepsilon & \sin\delta\cos\varepsilon & -l_2\cos\delta\sin\varepsilon \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{43} = \begin{pmatrix} \cos\mu\cos\nu & -\sin\mu & -\cos\mu\sin\nu & Rx_3 \\ \sin\mu\cos\nu & \cos\mu & -\sin\mu\sin\nu & Ry_3 \\ \sin\nu & 0 & \cos\nu & Rz_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad M_{54} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A robot-head in the image is characterized by an affine transformation that maps the prototype image onto a certain portion of this image. The real distance between the two robot heads is calculated as follows: the prototype head's image has four vertex: A, B, C, D and was taken with a distance ρ (when the picture was originally taken). The affine transformation maps this image into a rhomboid with vertex A', B', C' and D'. As the visual area decreases in a quadratic way with the distance, if the camera has no distortion, the Rx_3 coordinate of the observed robot's head in the $\{\hat{i}_3\ \hat{j}_3\ \hat{k}_3\}$ axis system can be calculated as:

$$Rx_3 = \rho\sqrt{\frac{prototype\ image\ area}{mapped\ area}} = \rho\sqrt{\frac{d(AB)\times d(BC)}{\det(\overrightarrow{A'B'}\quad \overrightarrow{B'C'})}}$$

Where ρ is the distance between the camera and the prototype head (at the acquisition time). If the horizontal angle of view of the camera is *Wu* and the vertical one is *Wv*, the camera resolution is *Mu* (horizontal) x *Mv* (vertical), and the head image's center is in the image position (u,v), then Ry_3 and Rz_3 can be calculated as:

$$\mu = Wu\frac{Mu/2-u}{Mu},\ \nu = Wv\frac{Mv/2-v}{Mv};\ Ry_3 = Rx_3 tg^{-1}(\mu)\ ,\ Rz_3 = Rx_3 tg^{-1}(\nu).$$

4 Experimental Methodology and Results

The critical part of the here-proposed gaze determination system is the detection of the robot-heads. Therefore in this article results of this sub system are reported. In a future work we are going to report experimental results of the whole system.

The robot-head detection system was implemented in the AIBO ERS-7. The subsampled scale-space is built from the original AIBO images (208x160 pixels). Using these small images speeds up the calculations, but the use of a small initial Gaussian (σ=0.7) for building the scale-space makes the computation of interest points very noisy. This is the reason why the here-proposed parabolic interpolation and additional verification stages must be used. The SIFT points and descriptors calculation takes between 1.05 seconds and 1.25 seconds, depending on the number of objects under observation. The matching voting and transformation calculation takes around 30 milliseconds for each prototype head analyzed.

Robot-head detection experiments using real-world images were performed. In all of these experiments the 16 prototypes of robot player 1 were employed (see figure 2). These prototypes (around 100x100 pixels) are stored in the flash memory as BMP files. A database consisting on 39 test images taken on a four-legged soccer field was built. In these images robot "1" appears 25 times, and other robots appear 9 times. 10 images contained no robot. Currently this database is been expanded to be made public, together with the robot prototypes database. In table 2 are summarized the obtained results. If we consider full detections, in which both, the robot-head pose as well as the robot identity is detected, a detection rate of 68% is obtained. When we considered partial detections, i.e. only the robot identity is determined, a detection rate of 12% is obtained. The combined detection rate is 80%. At the same the number of false positives is very low, just 6 in 39 images. These figures are very good,

because when processing video sequences, the opponent or teammates robots are seen in several consecutive frames. Therefore, a detection rate of 80% in single images should be high enough for detecting the robot-head in few frames.

Although more intensive experiments should be performed for characterizing our system, we believe that these preliminary experiments show the high potential of the proposed methodology, as a way of achieving player recognition and gaze estimation. The SIFT descriptors are not based in color information; therefore they are complementary to existing vision systems employed in the RoboCup leagues. A mixed SIFT and color-based vision system could be employed in the four-legged league if the SIFT computation time could shortened.

Table 2. Robot-head detection of robot #1 (only robot #1 prototype were employed)

Full detections (head + identifier number)	17/25	68%
Partial detections (only the identifier number)	3/25	12%
Full + partial detections	20/25	80%
Number of false detections in 39 images		6

References

1. D. G. Lowe, Distinctive Image Features from Scale-Invariant Keypoints, *Int. Journal of Computer Vision*, 60 (2): 91-110, Nov. 2004.
2. M. Brown and D. G. Lowe, Invariant Features from Interest Point Groups, *British Machine Vision Conference* - BMVC 2002, 656 – 665, Cardiff, Wales, Sept. 2002.
3. V. Bakic, G. Stockman, Real-time Tracking of Face Features and Gaze Direction Determination, *4th IEEE Workshop on Applications of Computer Vision* – WACV'98, 256 – 257, Princeton, USA, Oct. 19 - 21, 1998.
4. A. Perez, M.L. Cordoba, A. Garcia, R. Mendez, M.L. Munoz, J.L. Pedraza, F. Sanchez, A Precise Eye-Gaze Detection and Tracking System, *11th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision* - WSCG'2003, Plzen - Bory, Czech Republic 3-7 February 2003.
5. C. Harris and M. Stephens, A combined corner and edge detector, *4th Alvey Vision Conf.*, 147-151, Manchester, UK, 1988.
6. F. Schaffalitzky and A. Zisserman, Automated location matching in movies, *Computer Vision and Image Understanding* Vol. 92, Issue 2-3, 236 – 264, Nov./Dec. 2003.
7. K. Mikolajczyk and C. Schmid, Scale & Affine Invariant Interest Point Detectors, *Int. Journal of Computer Vision*, 60 (1): 63 - 96, Oct. 2004.
8. Q. Ji and X. Yang, Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance, *Real-Time Imaging*, 8, 357-377 (2002).
9. T. Ohno, N. Mukawa and A. Yoshikawa, FreeGaze: A Gaze Tracking System for Everyday Gaze Interaction, *Symposium on Eye Tracking Research and Applications*, 125-132, 2002.
10. J. Beis and D.G. Lowe, Shape Indexing Using Approximate Nearest-Neighbor Search in High-Dimensional Spaces, *IEEE Conf. Comp. Vision Patt. Recog,* 1000-1006, 1997.
11. D.G. Lowe, Local Features View Clustering for 3D Object Recognition, *Proc. of the IEEE Conf. on Comp. Vision and Patt. Recog*., 682 – 688, Hawai, Dic. 2001.