Abstract— The paper presents a mobile robot localization system that integrates Monte-Carlo Localization (MCL) with an active action-selection approach based on an aliasing map. The main novelties of the approach are: the off-line evaluation of the perceptual aliasing of the environment; the use of this knowledge to actively perform the localization processes; the use of an improved SIFT feature extractor to aliasing map evaluation and to measure image similarity. Results, obtained in a real scenario using a real robot, show improved performances in the number of steps needed to correctly localize the robot and in the localization error, compared with the classic MCL approach. Also better performances in computational time due to improvements in the vision system are shown.

I. INTRODUCTION

Localization is one of the most basic abilities for a mobile robot. A robot needs some kind of representation of the environment to self-localize, i.e., to be able to determine its own position in the environment. Explicit or geometric representations are based either on maps of free spaces in the environment (using, for instance, grid maps or polygonal-based representations) or on maps with locations of distinct observable objects (i.e., landmarks). Both methods rely on the assumption that geometric information (shape and position of obstacles, landmarks, etc.) can be extracted from robot's sensor readings. However, the transformation of sensor readings into geometric information is, in general, complex and prone to errors, increasing the difficulty of the localization problem [4].

In general, the localization model is implemented as a background process, i.e., localization is performed while the robot is involved in another task. However, when the robot is uncertain about its position it makes little sense to continue with the navigation task and it is more advisable to first reduce the uncertainty in the robot position and then continue with the navigation. Therefore, sometimes robot's actions have to be determined by localization criteria and not by navigation related ones. The question in such cases is how to determine the appropriateness of actions as far as localization is concerned.

In the context of mobile robot localization, actively controlling a robot is particularly beneficial when the environment is characterized by relatively few features that do not enable a robot to unambiguously determine its location. This is the case in many office environments. For example, corridors and offices often look similar to the perception system of a mobile robot.

Approaches were presented to solve this kind of problem dealing in particular with the concept of entropy minimization [1, 3]. A first limitation of these approaches arises from the algorithmic complexity of the entropy prediction. With not complex environments, algorithmic tricks can make the computation of entropy feasible, but more research is needed to scale such approaches to very large environments. A second limitation arises from the greediness of action selection. In principle, the problem of optimal exploration is NP hard, so situations where a greedy solution will fail exist. However, better results than the passive counterpart are obtained even if with a very high computational cost. These approaches use often range sensors and a previously known map of the environment.

On the contrary, the approach described in this paper has a very low computational cost and, in particular, the active action selection is based only on a look up table evaluated off-line, given a database of images (visual map) of the environment. A preliminary visual tour is used to take some pictures of the environment and store their features and positions in a reference image database. The map is a graph of nodes covering the two-dimensional environment and where each node contains the features extracted from the image at the respective position.

In this paper we propose a mobile robot localization system that integrates Monte-Carlo Localization with an active action-selection approach based on vision and on an aliasing map. The key idea is to evaluate off line the perceptual aliasing of every point of the map and use this information to bring particles and robot in a less aliased point of the map so to disambiguate robot position and to obtain faster global robot localization. Moreover, the perceptual system uses an improved version of the Lowe's SIFT feature extractor [8].

The appropriateness of our approach is demonstrated empirically using a mobile robot in a structured office environment and comparisons with the classical MCL approach are also presented to demonstrate the better performances and the lower computational cost of our algorithm.

The concept of aliasing maps is introduced in the next section, while a detailed description of the proposed algorithm is given in Section III. Section IV presents details of the vision system, while results and conclusions are reported in Section V and VI, respectively.
II. ALIASING MAPS

The main idea of the proposed approach is to evaluate off-line the perceptual aliasing (i.e., different places that perceptually appear the same) of the environment and to use this knowledge to perform localization processes faster and better. In particular, we exploit this knowledge to bring the robot to a less aliased position in the environment, thus disambiguating robot belief and allowing a faster and more accurate localization.

The approach here proposed for off-line perceptual aliasing evaluation over a known map of images (a database of geo-referenced images), is also general to every kind and number of range sensors (e.g., laser or sonar).

An aliasing map (Fig. 1) is defined by the function \( \text{Aliasing}(x, y) \) that returns the similarity of a point with all other points of the map. To simplify, in our approach we do not consider rotations. This assumption is valid whenever we consider a system able to measure range distances at 360 degrees around the robot, or when an omnidirectional vision sensor is used.

A graph is defined over a map of the environment and a sensor reading is assigned to each node of the map. Sensor readings are images taken from the node position and stored in an image database. In particular, we adopt an omnidirectional image as sensor reading. The aliasing in each node is measured as the number of sensor readings in different nodes that are similar (above a threshold) to the current one. Results are stored in the matrix \( \text{AliasingMatrix} \) whose number of rows and columns is equal to the number of graph nodes and sensors used, respectively.

If, after the visual tour, there are nodes of the map without a reference image in the database, they are considered as high aliased nodes. The graph nodes are then interpolated using a bi-cubic interpolation algorithm to obtain a fine grid aliasing map.

The approach can be generalized by taking into account also the rotation \( \sigma \) resulting in the function \( \text{Aliasing}(x, y, \sigma) \). This is the case of directional sensors as, for example, stereo vision.

The aliasing map is integrated with an MCL approach to active localize a mobile robot. We assume that robot knows its initial orientation with a certain error. The direction of the best robot’s action is given by the sum of all the minimum aliasing directions of each particle that allow to bring robot and particles to a minimum aliasing region of the environment.

If we consider a square and we assume to do not care about rotation, a point in a corner of a square environment gives the same sensor readings of other corner points. The only not aliased point of the environment is the center of the square (Fig. 1a). A real aliasing map of an indoor environment is shown in Fig. 1b.

III. MCL IMPROVEMENTS

The Monte-Carlo-Localization method uses a particle filter to approximate a probability function for estimating the robot position.

The particle distribution is initialized either as a uniform distribution over the entire environment or according to some prior knowledge about the position. Using a probabilistic motion model, after each robot motion, the particles are moved accordingly, introducing some randomness. Periodically, observations from some external sensors, are used to evaluate the position hypotheses represented by the particles. This is done by associating a weight to each particle that reflects the probability of the current observation at the particle’s location and by re-sampling the particles according to the distribution given by the weighted particles. MCL and similar approaches also appear in the literature as particle filters, condensation algorithms, sampling importance resampling (SIR) filters, etc.

The main novelty of the proposed approach is that the aliasing map is also used to evaluate the direction that brings every particle to a minimum aliasing position in the environment, and so to actively localize the robot.

Fig. 1. a) The aliasing map of a square environment: the minimum is in the center. b) A real aliasing map of an indoor environment.
over the map, given movements and observations at time $k$.

The main goal of this kind of MCL is to track a variable of interest, typically a not Gaussian and multi-modal probability density function, as it evolves during time. This method is based on the discrete representation of the probability density function of interest; every sample is called particle. To track the time evolution of the probability density function every particle is updated according to system dynamics. At every step some observations are available and they act as constraints for the future evolution of the particle population. A weight proportional to the probability of representing the true variable is associated to each particle. This weight is a function of the sensor model. At every time step the tracked variable can be estimated as the weighted mean of the particle population.

In our particular case the variable of interest is the position of the robot. Every particle represents a possible position over the map. A weight proportional to the probability of representing the true robot position is associated to each particle; the particle weight is function of observations (i.e., image similarity between the current observation and the reference image nearest to the current particle, in the real robot example).

Particle filters algorithms are divided in two phases: prediction and update. After every action taken by the robot a model of this movement is applied to every particle. Each weight is updated on the basis of the observation taken by the robot on the new position.

As a result of the observation, each weight is updated, proportional to the probability of representing the true position of the robot. Every particle represents a possible position of the robot. Each weight is updated on the basis of the observation taken by the robot on the new position.

The computational cost of the action selection is very low. Indeed only a differential evaluation for each particle is performed and this computation can be performed offline and stored in a look up table.

The approach to the re-sampling that we adopted can be divided in four steps, each characterized by a specific operation taken over the population.

The first step is known in literature as "selection with replacement". Each particle has a probability proportional to its weight of being regenerated a casual number of times into the new population. In this way the elements with a high weight will be repeated a higher number of times than those with a low weight. During the application of this re-sampling technique the number of particles remains constant, but the new population should represent in a better way the true position of the robot.

In the second step the population is resized (i.e., the number of particles decreases at every step), so that the computational load can be optimized to the localization process. We introduce an index to describe the quality of the localization process. This index represents the dispersion of particles on the map area.

The third step of the re-sampling process operates a control on the position of each particle to minimize the influence of any element of the population whose position will not be in the admissible area of the map. In the case that a particle has coordinates external to the admissible map region, it will be eliminated and then regenerated with a small weight in the map.

Finally, a certain percentage of the total number of particles is distributed over the map, according to a probability distribution that is the negative of the aliasing direction. These features have been widely used in the

IV. THE IMPROVED SIFT APPROACH

The Scale Invariant Feature Transform (SIFT), developed by Lowe [7, 8], is invariant to image translation, scaling and rotation. SIFT features are also partially invariant to illumination changes and affine 3D projection. These features have been widely used in the

Position estimation

$$\tilde{x}_{k+1} = \sum_{i=1}^{n\text{Particles}} w_i' x_{i,k+1}$$

Action Selection

$$v_p = \text{grad}(\text{AliasingMatrix}(x_{k+1}))$$

$$V_r = \sum_{i=1}^{n\text{Particles}} w_i' v_p$$

$$k = k + 1$$

where $xMax$ and $yMax$ are map limits along $x$ and $y$ axes and $\hat{x}$ is the estimated robot position (weighted mean of particle positions). The number of steps of the localization algorithm is represented by $k$.

We assume that the robot knows its initial orientation with a certain error. $V_r$ is the direction of the best robot’s action, given by the sum of the minimum aliasing direction $v_p$ of each particle that allows bringing robot and particles to a minimum aliasing region of the environment.

The approach to the re-sampling that we adopted can be divided in four steps, each characterized by a specific operation taken over the population.

The first step is known in literature as "selection with replacement". Each particle has a probability proportional to its weight of being regenerated a casual number of times into the new population. In this way the elements with a high weight will be repeated a higher number of times than those with a low weight. During the application of this re-sampling technique the number of particles remains constant, but the new population should represent in a better way the true position of the robot.

In the second step the population is resized (i.e., the number of particles decreases at every step), so that the computational load can be optimized to the localization process. We introduce an index to describe the quality of the localization process. This index represents the dispersion of particles on the map area.

The third step of the re-sampling process operates a control on the position of each particle to minimize the influence of any element of the population whose position will not be in the admissible area of the map. In the case that a particle has coordinates external to the admissible map region, it will be eliminated and then regenerated with a small weight in the map.

Finally, a certain percentage of the total number of particles is distributed over the map, according to a probability distribution that is the negative of the aliasing direction.
robot localization field as well as in many other computer vision fields.

The SIFT algorithm has the following four major stages.

1. Detection of scale-space extremes: the first stage searches over scale space using a Difference of Gaussian (DoG) function to identify potential interest points.

2. Localization of key points: location and scale of each candidate point are determined and key points are selected on the basis of measures of stability.

3. Assignment of key point orientation: one or more orientations are assigned to each key point on the basis of local image gradients.

4. Generation of key point descriptors: a descriptor is generated for each key point from local image gradients information at the scale found in stage 2.

Let us briefly clarify the first stage. For each octave in the scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images. Adjacent Gaussian images are subtracted to produce the DoG images. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process is repeated. For a more detailed discussion of the key point generation and factors involved see [8].

While invariant to scale and rotation, and robust to other image transforms, the SIFT feature description of an image is typically large and slow to compute.

Consequently we compute the image similarity in the innovation term using a reduced and optimized SIFT approach with 64 feature descriptors and we introduced time saving improvements by the following two main steps: adaptation of SIFT parameters to each sub-image (Fig.2), extraction of a fixed number of key points.

In particular, the number of scales of original image is defined according to its dimensions and thus in some cases not all SIFT scales are necessary to be computed. The following threshold value \( (Tr) \) is also computed to define the contrast threshold value of the SIFT algorithm:

\[
Tr = k \cdot \frac{\sum_{i,j=0}^{DimX \times DimY} [f(x_i,y_j) - \bar{I}(x_i,y_j)]}{DimX \cdot DimY}
\]

where \( k \) is a scale factor, \( DimX \) and \( DimY \) are the \( x \) and \( y \) image dimensions, \( f(x,y) \) is the intensity of the gray level image and \( \bar{I}(x,y) \) is the medium intensity value of the processed image.

In the Lowe’s SIFT implementation the contrast threshold is statically defined and low contrast key points are rejected because they are sensitive to noise. In our implementation this threshold is computed for each sub-image, sometime avoiding at all the time-consuming feature extraction process and, in any case, allowing to deal with different lighting conditions.

We also want to reduce the number of key points and their corresponding extraction and matching time, while maintaining the same descriptor for each key point.

In the classical SIFT approach, key points are detected by testing each value in the DoG at each scale with the 8 surrounding values of the same scale as well as with 9 neighboring values in the scale above and 9 neighboring values in the scale below. The first and last DoG scales are not examined. This means \( 26 \times m \times n \) comparisons for a DoG of size \( m \times n \), taking into consideration that points around a given border of each DoG are not included in the key point detection. Since SIFT establishes multiple scales in each octave, the above analysis is applied several times to each scale in each octave. Each octave has one quarter of the pixels of the previous one, so that key point detection in lower octaves requires more time than in higher ones. We aim to modify this exhaustive search into a sample based one.

In the approach we propose, the number of key points can be defined in advance. The process of finding the key points continues iteratively without the need for sequentially going through the whole scale space. This involves two phases. In the first phase the scale space is randomly searched for local extremes. The random search is followed by an update phase only when the local extreme is more likely to be found. The theory behind this approach is mainly based on the assumption that local extreme points are located in a blob region, i.e. smooth wide two dimensional hills or valleys. In other words, blobs are regions in the image that are either significantly brighter or significantly darker than their surroundings. A local extreme cannot be located on a flat region and can hardly be found near it. Spikes, that is rapidly changing narrow regions, are another possible location of local extremes, but since the scale space structure involves multiple smoothing operations on the image, only information on the coarse scale remains and the spikes are filtered out.

With the above assumption, our search mechanism have to deal with only two cases when searching for a local extreme:

1) when the system detects a blob region an update phase handles the search for the position of the local extreme.
extreme in that region (the search ends either when the local extreme is found or after a given number of trials);

2) when the system detects a non-blob region the result of the search in this area is ignored and the search is started somewhere else.

More in particular, we first initialize, for each scale, a set of candidate key points (samples) by selecting random couples of numbers, each representing the coordinates of one point in the image. The samples are then verified and only those that have a value above the given threshold will be considered stable key points. This reflects our assumption that a value above the threshold is most probably a point that lies in a blob. A similar approach to this one was also introduced in [9].

Because the number of matched key points is defined in advance the computation time will result proportional to that number.

Algorithm tuning and early results have been obtained using a data set of outdoor images where we artificially performed lighting and contrast variation to every image.

Here below a comparison of the proposed feature extraction process with the classical Lowe’s SIFT is reported. Fig. 3 reports an example of feature matching between an original and a dark image.

Results showed in Table I demonstrate the better results obtained using the proposed approach. The key result of this experimental comparison is that the computational time for feature extraction is lower than the standard SIFT implementation despite of the average number of matched features is drastically higher using our method.

<table>
<thead>
<tr>
<th>Matched features</th>
<th>Computational time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowe’s SIFT</td>
<td>13</td>
</tr>
<tr>
<td>Improved SIFT</td>
<td>94</td>
</tr>
</tbody>
</table>

These feature descriptors are then used to represent the environment appearance at the respective position and stored in a map. A preliminary visual tour is used to take some pictures of the environment and store their features and positions in a reference image database. The map is a graph of nodes, covering the two-dimensional environment, where each node contains the features extracted from the image at the respective position. Global image similarity is computed by dividing the total number of features matched between the two images by the number of features extracted from the reference images.

V. RESULTS

Results have been obtained using an Active Media Pioneer 3 DX robot, equipped with an omnidirectional camera, in an area of our department.

This robot configuration, using omnidirectional vision, provides for the hypothesis of omnidirectional sensors of the proposed approach.

We performed a run with a length of approximately 40 m. By manually marking the robot position at specific reference points and interpolating between these for odometry correction, the absolute robot position (ground truth) is known very well along the entire path. For the visual localization a reference image database with 250 images, taken from the corridor and from offices, was built.

Table II. Absolute position-error for standard MCL and improved MCL, measured after 20 robot steps

<table>
<thead>
<tr>
<th>Error (mm)</th>
<th>Standard MCL</th>
<th>Improved MCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>After 20 steps</td>
<td>510</td>
<td>430</td>
</tr>
</tbody>
</table>

Fig. 4 Map of our department. The grey part is the high aliased area that was used in the robot localization tests described in this paper.
The used framework is a mixed Matlab – C++ mobile robotics platform that well suits to research and educational purposes. Further details about the framework can be found in [2].

The testing environment is an area of our department (Fig. 4), a structured office environment of about 200 m² with high perceptual aliasing. The purpose of this final test was to evaluate performance and robustness of the algorithm for the global localization in a real environment. The aliasing map is evaluated off line, taking about 3 minutes, from the map of the environment, before starting global localization experiments. During our test we started the algorithm using 300 particles and the robot was able to localize itself in about 20 steps with a mean error lower than 50 cm.

Because of the partially random nature of classic MCL, we executed 10 runs over the same area to obtain significant results. The absolute position-error for standard MCL and improved MCL, measured after 20 robot steps, is shown in Table II.

VI. CONCLUSIONS

The paper advocates a new active vision based approach to mobile robot localization. In active localization the robot controls its various effectors to most efficiently localize itself. In essence, actions are generated by maximizing the expected decrease of uncertainty, measured by perceptual aliasing.

The key result of the experimental comparison is that the efficiency of localization is increased when actions are selected with the aim of minimizing global aliasing. In some cases, the action selection component enabled a robot to localize itself where the passive counterpart failed. Also, the active action selection procedure has a low computational cost.

The main drawback of this approach is on its application to dynamic environments. Actually the off-line aliasing map evaluation does not guarantee a reconfiguration of algorithm parameters when dynamics events happen (e.g., people moving around the robot).

The main assumption behind our approach is that we assume robot initial orientation known. This is not a really strong assumption because we can know initial absolute orientation using a compass. Furthermore we experienced particle filtering robustness to initial orientation errors and so the localization system only needs an initial rough estimation of robot absolute orientation.

Besides, in these experiments we tested a practical idea to improve and speed up the SIFT technique. The number of matched key points can be defined in advance and the computation time is proportional to that number. We also introduced the idea of parameter adaptation to avoid feature extraction from uniform regions of an image. We applied these strategies to a data set of outdoor images and we demonstrated that the approach is suitable since we need to deal with lightness changes. Even if results are preliminary, the general idea applied to mobile robotics gives desirable performances. The approach can be generally applied to any similar problem and we plan to perform mobile robot localization tests in outdoor environments.

Finally, we also planned to test this approach together with sensor fusion between range and vision sensors.

REFERENCES