A particle-filter-based self-localization method using invariant features as visual information

Jesús Martínez-Gómez, Alejando Jiménez-Picazo, Ismael García-Varea
University of Castilla-La Mancha, Spain
{jesus_martinez,ajimenez,ivarea}@dsi.uclm.es

Abstract

This article presents a new approach to robot localization in indoor environments. The system presents a Monte Carlo method using SIFT for the visual step. The scope of the system is indoor environments where no artificial landmarks are necessary. The complete pose $< x, y, \theta >$ can be obtained. Results obtained in the RobotVision@ImageCLEF competition proved the goodness of the algorithm.

Categories and Subject Descriptors

H.4 [**Cross-Language Retrieval in Image Collections**]: H.4.4 Robot Vision

General Terms

Evaluation, Image Classification, Robot Localization

Keywords

Mobile robots, image processing, localization, particle filters, Scale-Invariant Features Transform

1 Introduction

Self-localization is one of the harder problems in mobile robot research. Intelligent robots need to perform different tasks depending on their own pose. If a robot's pose is not well estimated, the behaviour obtained will not be correct.

Different approaches have been successfully implemented to deal with real-world problems (noise, uncertainty, low-quality images) [3, 4, 11, 9]. Most of these approaches combine the information obtained from two different sources: perception and odometry. Perception uses all the robot's sensors to retrieve information from the environment. Main sensors are distance sensors and vision cameras. Odometry can be used to estimate the pose of the robot using the last pose and the set of movements the robot has performed so far. Both sources (perception and odometry) are noisy and with a high uncertainty, so the combination of these sources of information must be performed appropriately.

The RoboCup[1] competition is a good scenario where different solutions have been proposed over the last few years. These proposals use natural or artificial landmarks in the environment to estimate the robot's pose (goals, beacons and field lines). Within these controlled environments, the absolute position of these landmarks does not vary and the robot's pose can be obtained by estimating distances and orientation in relation to these elements. Other environments, such as

[1] http://www.robocup.org/

that proposed for the RobotVision@ImageCLEF, do not have natural landmarks to be used for robot pose estimation. These dynamic environments force us to use other techniques without the use of landmarks. These techniques are commonly based on the information obtained from distance sensors, such as sonar or laser.

The RobotVision@ImageCLEF task addresses the challenge of localization of a mobile robot using only visual information. Neither distance sensors nor odometry information is provided with the final test sequence. This is why this article is focused on image processing and how to estimate a robot's pose using this processing.

The approach presented here carries out localization by using Scale-invariant feature transform[6] for the visual step. The principles of the particle-filter-based Monte Carlo [2] method are applied with some variations. The combination of this localization method with a robust image processing algorithm allows the robot to reduce the uncertainty about its pose when captured images have a good resolution. The algorithm keeps information about the robot's pose even when the quality of the images decreases.

Different experiments using training image sequences and the final validation test sequence have been carried out to evaluate our proposal. These experiments were performed for the RobotVision@ImageCLEF task (see [1] for more information).

The article is organized as follows: SIFT techniques are outlined in Section 2; we describe the Monte Carlo algorithm in Section 3 and in Section 4 we explain the approach we adopted for the task. Section 5 describes the experiments performed and the results obtained. Finally, the conclusions and areas for future work are given in Section 6.

2 Scale-Invariant Feature Transform

Scale-Invariant Feature Transform[7] (SIFT) is a computer vision algorithm, developed to detect key features in images. This algorithm was developed and published by David Lowe in 1999, and now there are different versions and variations available.

The main idea of the algorithm is to apply different transformations and study the points of the image which are invariant under these transformations. These extracted points can be used to perform object recognition by carrying out a matching between images representing the same object or scenario.

Features extracted are invariant to image scale and rotation, and they are also robust to noise and changes in viewpoint. An important characteristic of systems developed to perform object recognition using SIFT is that they are robust to partial object occlusion.

The SIFT implementation used was developed by R. Hess, from the Oregon State University [2]

2.1 Algorithm

Different steps are necessary to extract invariant points.

First, the image is convolved with Gaussian filters at different scales. Key locations are selected at maxima and minima of difference of these filters. After this step, too many candidates are obtained. A second step discards low contrast key points by interpolating the data near each candidate point. The interpolation is performed using the Taylor expansion[10] of the difference of the Gaussian scale-space function. Finally, the algorithm eliminates candidates not located on edges.

2.2 Problems and restrictions

The main problems of using this algorithm are the long execution time and the high demand on memory. The execution time necessary to extract features from a 309 x 240 image (size used for the task) is about 0.5 seconds using a 2.13GHz dual core processor[3]. The matching between

[2] http://web.engr.oregonstate.edu/~hess/

[3] All the times reported along the paper were obtained using that machine

two extracted feature sets takes 0.1 seconds using a nearest-neighbour search in high-dimensional spaces.

If we denote T_{ext} as the time necessary to extract the features and T_{mat} as the time to perform the matching, the processing time T_{proc} can be defined using Eq.1, where n is the number of images to compare with.

$$T_{proc} = T_{ext} + T_{mat} * n \tag{1}$$

The main problem of the execution time is that it depends on the number of images to compare with. For the RobotVision@ImageCLEF task, where the final training set contains 1690 frames, the time needed to classify one image using our computer will be $0.5 + 1690 * 0.1 \approx 2.82$ minutes.

3 Monte Carlo localization method

Different formal methods for the localization problem have been proposed over the last few years. Nowadays, two methods are widely used and most localization systems are defined using their principles. These methods are particle filters and (Extended) Kalman filters [9].

Kalman filters are a robust method that works properly under optimal conditions. Its execution time and demand on space are small, but an important drawback is that it is necessary to know the robot's initial pose. Moreover, this method presents problems when the uncertainty about the pose increases, or when the robot can suddenly be kidnapped (such as in the RoboCup environment).

Particle filters spread particles over the environment. Each one of these particles represents a robot's pose $< x, y, \theta >$. An iterative process applies an odometry step to each one of these particles and performs the weighting step. The evaluation of each particle is obtained by processing the information sensed from the environment, using the camera or distance sensors. After all the particles have been pondered, a sampling step is applied. A new population of particles is obtained from the last iteration, and the particles are sampled with replacement using their goodness value as selection probability. Best candidates should be duplicated, and worst particles should disappear from the environment. After some iterations, the algorithm will converge and all the particles will be around the real robot's pose, as can be observed in Fig. 1.

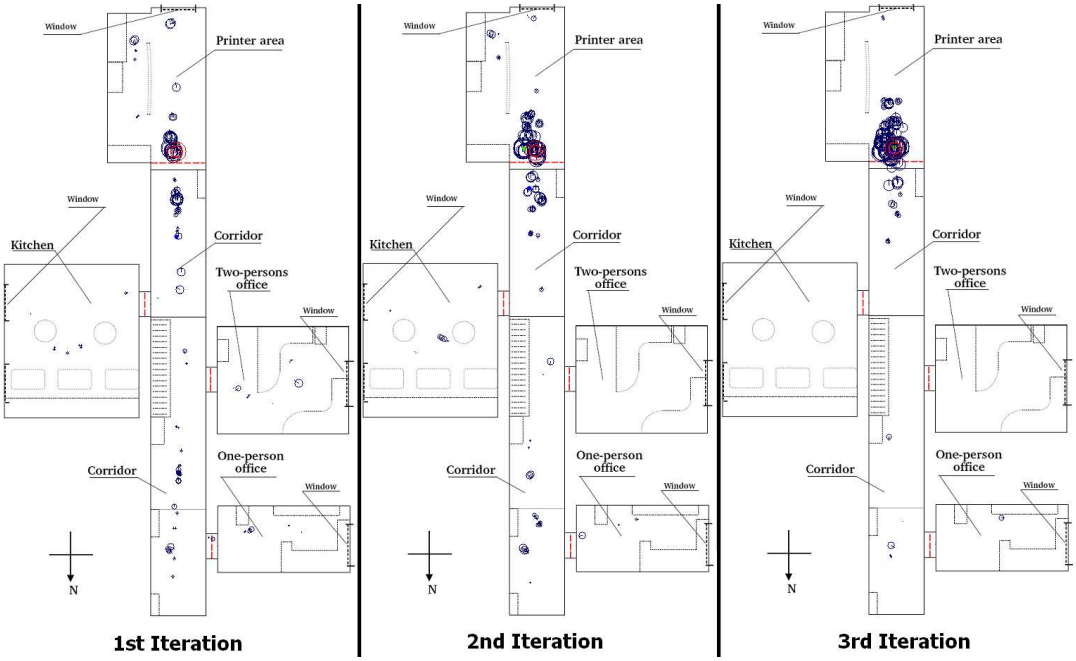

Figure 1: An example of particle convergence after three iterations of the MC localization algorithm

The main advantage of the algorithm is that uncertainty about pose estimation can be kept throughout the process. If the robot detects that it is close to an internal door, particles will be distributed at different parts in the environment. This situation can be observed in Fig. 2.

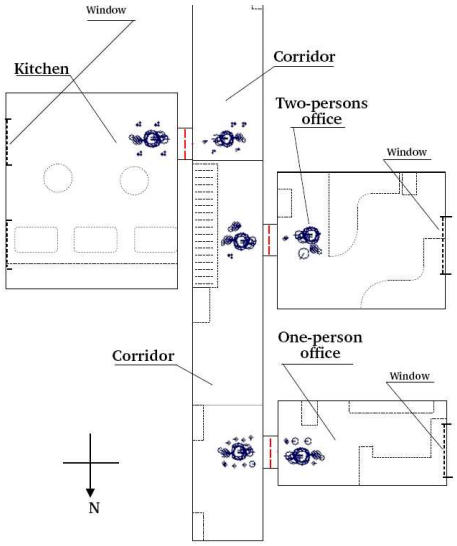

Figure 2: Particle distribution when the robot detects an internal door

3.1 Process

The process of a particle-filter-based localization method consists of a prediction and an update phase. These phases are repeated iteratively until a specific stop condition is reached. For instance, the stop condition can be the end of a test frame sequence.

3.1.1 Prediction Phase

In this step we start with the set of particles from the last iteration. We apply the movement model to each one of these particles. In order to apply this phase correctly, it will be necessary to have an accurate odometry model.

3.1.2 Update Phase

In this phase, the information obtained from sensors is taken into account. Each particle is weighted using the likelihood of obtaining the measurement from the pose it represents. After all the particles are weighted, the resampling step is applied. This resampling step generates a new set of particles. The weight of each particle is used as the selection probability for the set of particles to be part of the next iteration of the algorithm.

4 SIMD approach

Our proposal for the RobotVision@ImageCLEF task was to develop a localization method using particle-filter principles and SIFT for the update phase. We used the KHL-IDOL2 [8] available set of frames to train the system. The information retrieved after applying SIFT to training frames was the complete pose $< x, y, \theta >$, the correctly-classified room and the set of invariant points extracted from the frame.

The definition of these points was stored in data files. Each training frame has an associated file with all the obtained SIFT points. This allows us not to have to recompute SIFT points when

an image matching process has to be performed, with the associated saving in computing time. These files also stored the pose and the room corresponding to the image. By reading the complete file sequence we can obtain all the training information.

We presented two approaches for two types of problems. The first one (obligatory track) corresponds to global topological localization, where the algorithm must provide information separately for each test image. For this problem, only visual processing (without any localization method) is applied. The second problem (optional track) corresponds to typical localization tasks, where image sequences are processed and the order is important. Visual processing is completed with a particle-filter-based method to develop the localization algorithm.

4.1 Obligatory Track - Image Processing

As mentioned above, only image processing is applied for this track. We uses a multi-classifier to label each image as an environment room. The first approach was to use only SIFT to classify each image, but the results obtained were not satisfactory due to the problems of SIFT with changing lighting conditions.

4.1.1 SIFT processing

SIFT processing is performed by extracting invariant features from the frame to be classified. These features are compared with the features extracted from the training sequence. The similarity between two frames is obtained by performing a matching between SIFT points extracted from them. This similarity is pondered as the percentage of common points.

After matching all the training frames with the current frame, we have to classify it. The current frame will be classified using the most similar training frames. This can be performed because all the training frames are correctly labelled with the room. We process only the best n training frames, using similarity to select the best candidates.

To classify a frame using the best n training frames, we compute the sum of all its weights. This summing is performed separately for the different rooms and finally, each room R_i will have an associated sum of weights sum_i. Each frame is classified as the room that obtained the highest sum of weights.

To show how this works, Tab. 1 presents the best 11 pondered training frames. With these pondered frames, we compute the sum of the weights separately for each room. After this sum is applied, we obtain the result that can be observed in Tab. 2. The current frame would be classified as Printer Area (PA).

Frame number	1	2	3	4	5	6	7	8	9	10	11
Weight	0.07	0.42	0.34	0.14	0.48	0.87	0.24	0.12	0.04	0.45	0.05
Room	CR	PA	PA	CR	PA	PA	CR	CR	BO	PA	BO

Table 1: Best 11 training frames with their associated weight and room

Room	PA	CR	BO	EO	KT
Sum of weights	2.56	0.57	0.09	0.00	0.00

Table 2: Separate sum of weights for the different rooms

In order to avoid classifying frames when the uncertainty about the robot's pose is high, we only classify frames when the maximum sum of weights for a room (2.56 in Tab. 2) is higher than 60% of the sum of all the weights (3.22 in Tab. 2).

A significant drawback to this reasoning is the execution time. It depends on the number of comparisons we perform. For the final RobotVision@ImageCLEF experiment, the training data sequence has 1690 frames. With a comparison time close to 0.18 secs and an SIFT extraction

time of 0.6 secs, it is necessary to spend more than 3 minutes to classify a single image. For the complete test sequence, the execution time was around 3.5 days of computing time.

Lighting changes are one of the main problems of using SIFT to localize the robot, because features extracted depend on the lighting conditions. During the development of the system, the results obtained were only acceptably good with training and test data acquired under the same lighting conditions. Due to this, we decided to add additional processing to improve the results.

4.1.2 Additional processing

SIFT was complemented with basic processing, based on line and square recognition. We apply a Hough[5] transform to study the distribution of the lines and squares obtained. Using images from the training sequences, we discovered some characteristics that could be detected using lines and squares.

These characteristics are unchanging sections from the environment that can be used as natural landmarks. Examples of these landmarks are the corridor ceiling, or the external door located in the printer area. We decided to use restrictive detection algorithms to avoid false positives. This processing was added to SIFT detection, and some examples of these detections can be observed in Fig. 3 and 4.

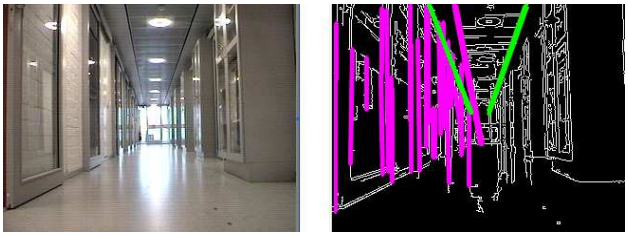

Figure 3: Corridor detection. Purple lines are candidates and green lines are the correct ones

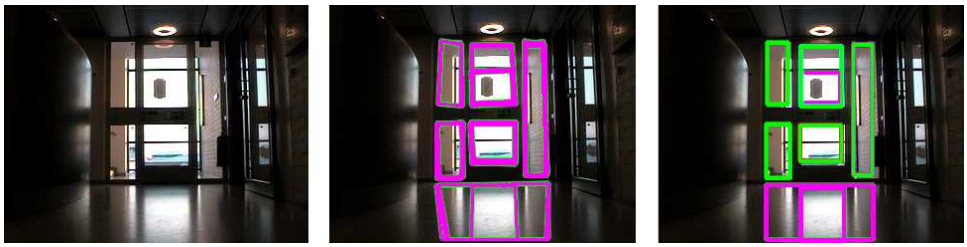

Figure 4: Printer area detection. Purple squares are candidates and greens squares are the correct ones

The main advantage of this processing is that the execution time on a typical current computer is very low (about 0.003 milliseconds).

4.1.3 Complete image processing

Taking into account the preliminary results we assume that the additional processing is more reliable than SIFT processing. Because of this, and the reduced execution time, we classify an image using the additional processing whenever possible. Otherwise, we extract the SIFT features and perform a comparison with the complete training sequence. This image processing techniques is used for the visual step of the algorithm developed for the optional track.

4.2 Optional Track

For the optional track, we propose an algorithm using particle-filter principles, and then we need to define the particle representation, the prediction step and the update step. It is worthy of note that despite the goal of the tracks being to classify each frame of the sequence into a specific room, our approach using a particle filter is also capable of providing the most reliable pose of the robot within the environment.

4.2.1 Particle representation

The particles used to localize the robot represent poses. A robot pose is defined using three parameters: x and y for position and θ for orientation. All these parameters have continuous numeric values. The limits for x and y values are the boundary of the environment. In this case, values for the x component are between 0 and 1500 cm. Limits for the y component are 0 and 2500 cm.

4.2.2 Prediction Phase

The information necessary to perform the prediction phase, related to the movement model, can be extracted from the training images. We assume that robot movement will be similar to that performed during the training (training frames were acquired while the robot was moving). Due to this, the robot's average velocity can be obtained from the difference between the poses of the training sequence. We obtain the average linear and angular velocity, which are defined in centimeters (and degrees) per frame.

In this step we add some uncertainty, commonly termed white noise, to the process. We modify the pose of each one of the particles using the movement model and a random variation. This variation depends on the particle's weight w_t. The idea is to apply a higher variation to the worst particles, in order to keep the best candidates with minor changes. We assume that the w_t value is between 0 and 1.

The robot's pose at instant t is denoted by $< x_t, y_t, \theta_t >$. Linear and angular velocities are denoted by v_l and v_a. A random value $rand$ between 0 and 1 was used to model the white noise. The maximum variations for x, y and θ are denoted by $maxvar_x$, $maxvar_y$ and $maxvar_\theta$.

To illustrate this process, we describe below the necessary processing for the prediction phase. The robot's estimated pose for the instant $t+1$, taking into account the pose estimated for the instant t, will be:

- $x'_{t+1} = x_t + v_l * sin(\theta_t)$
- $y'_{t+1} = y_t + v_l * cos(\theta_t)$
- $\theta'_{t+1} = \theta_t + v_a$

Maximum variations for x, y and θ components are computed using the particle's weight w_t.

- $maxvar'_x = maxvar_x * (1 - w_t)$
- $maxvar'_y = maxvar_y * (1 - w_t)$
- $maxvar'_\theta = maxvar_\theta * (1 - w_t)$

After applying the white noise, the estimated pose values will be:

- $x_{t+1} = x'_{t+1} + maxvar'_x - 2 * maxvar'_x * rand$
- $y_{t+1} = y'_{t+1} + maxvar'_y - 2 * maxvar'_y * rand$
- $\theta_{t+1} = \theta'_{t+1} + maxvar'_\theta - 2 * maxvar'_\theta * rand$

Movement applied to the particles will represent the distance and orientation variation we hope to obtain for the robot.

4.2.3 Update Phase

This phase obtains the weight for each particle (w_p). We use the information obtained from the robot's sensors and in this case, visual information is the only type that can be used. Thus, each particle is evaluated using the frame captured at the instant t, namely f_t.

The idea we propose is to use the image processing shown in section 4.1. In this case, we have to evaluate all the particles using the information extracted from the current frame, specially its SIFT points. Each particle is evaluated matching these points with those extracted from the frame taken from the pose the particle represents. The problem is that we only have SIFT points extracted from a small number of poses in the environment. These poses are the x, y and θ values of the images from the training sequence.

It will be necessary to search for SIFT points representing the nearest pose to the particle's pose. With bigger training sequences, the search space will be larger and the particle's evaluation will be more realistic. If a particle can only be evaluated with a distant SIFT representation, the evaluation will not be reliable.

The environment information is extracted during system training. While the training is being performed, we create a frame population. Each training frame (tf) stores the pose, the code of the room where the picture was taken and the SIFT points extracted from this image.

Algorithm 1 shows the general processing scheme for particle evaluation.

Algorithm 1 Particle Evaluation

```
1: sp_{f_t} ← Extract SIFT points from the current frame f_t
2: for (each particle p_i) do
3:    for (each training frame tf) do
4:       if (tf pose is the closest to p_i pose ) then
5:          match(sp_{f_t},sp_{tf});
6:          w_{p_i} ← % of common points;
7:       end if
8:    end for
9: end for
```

The execution time necessary to perform this step will be fully dependent on the number of particles, the size of the training sequence and the similarity between particles at instant t.

Additionally to SIFT processing, line and square detection can be used to detect the robot's position. If this process works, we will obtain the room where the robot is located and that information will be used to correctly classify the current frame.

Once all particles are weighted, we have to classify the frame. For particle-filter localization methods, this step corresponds to position estimation. Usually, the robot's position is estimated using particle information: x, y and weight w. Mean position is obtained as a pondered mean, taking into account particle weight:

$$\bar{x} = \sum_{i=0} x_i \times w_i, \;\; \bar{y} = \sum_{i=0} y_i \times w_i$$

It is possible to use the same reasoning and classify the frame as the room located at $< \bar{x}, \bar{y} >$ position. Problems arise when mean position does not corresponds to any room. To avoid this situation, we apply the same approach shown in section 4.1, adding the weights of all particles separately for each room.

If line and square detection classifies the frame with a room label, we have two possible scenarios: this room is the same as that obtained from all particles, or these rooms are different. If we obtain the same label, we assume that the algorithm is going well. Otherwise, the room will be labelled according to the result of line and square detection. In this situation, the particle population will be transformed to represent positions from the correct room, as will be explained in the next section.

4.2.4 Additional process

The first implementation of our system was a Monte Carlo localization method with the prediction and update phases as described above. This reasoning presented several problems, and it was necessary to modify the system, adding more processing.

The main problem was that some test frames were not represented by any training frame. This happens when there are no training frames from the pose where the test frame was taken or when SIFT matching fails. The consequence of both cases is that the weight w_{p_i} of the particles (or the % of points obtained in the matching process stated in Algorithm1) decreases continuously and they are spread over the environment. If this happens and we find false positives, the system will fail and the robot's location will not be correctly obtained. False positives occur due to noisy frames, luminance changes or when few SIFT points can be extracted from the test frame. With this situation, the system can converge at wrong environment positions. Reinitializing the particle population over the environment to recover from these situations did however obtain good results.

The first experiments added an extra step to the original Monte Carlo process. This step performs a population initialization when the best particle weight is below a certain threshold. This modification improved system behaviour when the localization failed and the particles converged at wrong environment positions. However, the system became unstable and the algorithm had problems to converge.

This situation can be observed in Fig. 5, where frame 11 fails and makes the system's accuracy decrease. All the particles are spread over the environment and the algorithm needs 8 new frames to recover from this situation. Fig. 5 shows the error for the position estimation. The estimation error increases quickly and affects the algorithm's performance. This happened when there was only one problematic frame with a controlled convergence situation.

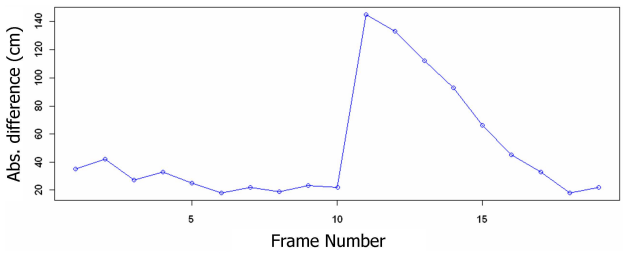

Figure 5: Problems encountered with particle re-initialization. Error for position estimation

This new modification should not affect the stability of the algorithm, increasing its vulnerability to problematic frames. These frames will be noisy pictures or those that have problems with SIFT matching.

To avoid such situations, the next step was to define a state for the algorithm based on its stability. The stability of the algorithm can be estimated by studying the particle poses. We store a number n of previous estimated robot poses. The process will be stable if the variation obtained for the x and y components of the last n pose estimations is sufficiently small. This happens when all the particles are close together without high variations between frames.

When the algorithm is stable, no population initializations are performed. Otherwise, the initialization will depend on the instability level. If the algorithm has been stable for the last few frames and suddenly it becomes unstable, initialization will be performed over a restricted area. This area will be a circumference centered at the most reliable robot position, obtained from previous iterations (with a stable process). The radius of this circumference depends on the instability level. This level is obtained with the best particle's weight for the last n iterations. Using this proposal, the algorithm becomes robust in those situations described above, with noisy and problematic pictures.

If the algorithm has been unstable for the last few frames and a new initialization has to be applied, all the particles will be spread over the environment.

All this processing is shown in Fig. 6.

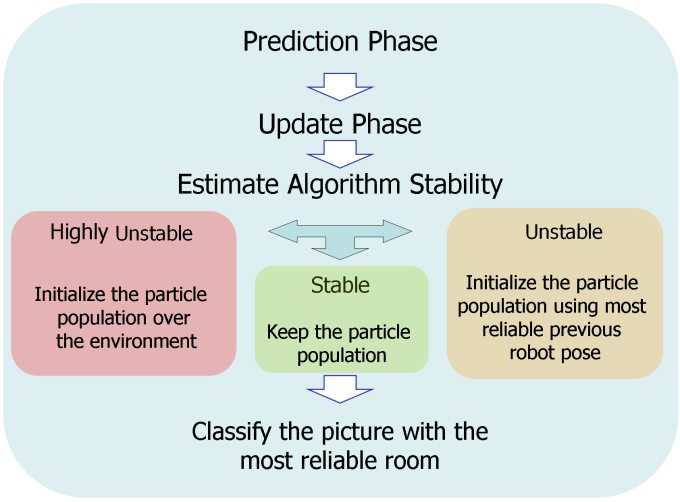

Figure 6: General Processing Scheme

In order to show how the system works, Fig. 7 presents three consecutive frames. For the first one, the algorithm was stable, and the robot's most probable pose was obtained (red circle). With the next frame, the system fails and the process becomes unstable. The algorithm performs a particle initialization for the third frame, but new particles are only created in the area denoted by the green circle. This circle is centered at the position $< x, y >$ obtained from the most reliable pose with the first frame.

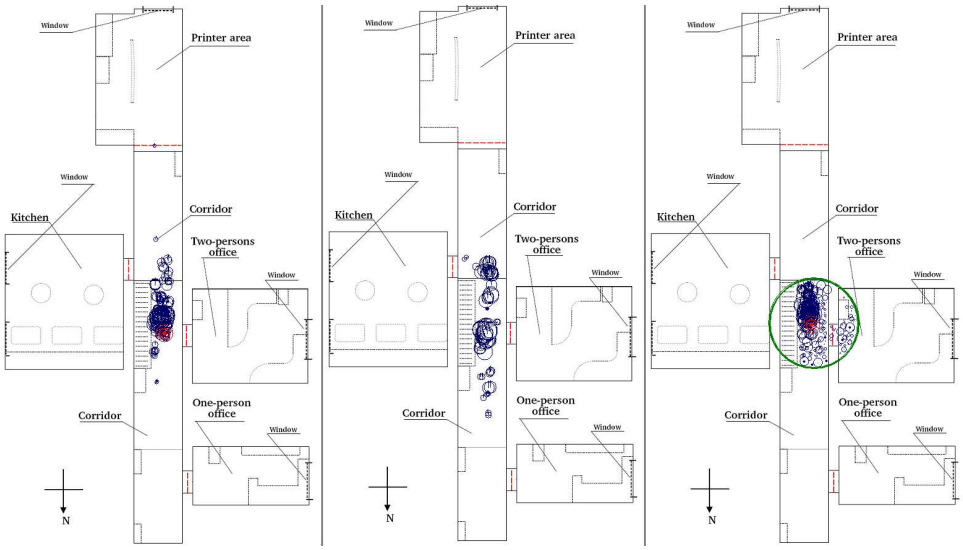

Figure 7: Particle initialization over a restricted area for three consecutive frames

In addition to this reasoning, and taking into account the CLEF task characteristics, we can add some processing for classification. The idea is to classify the images into rooms only when we are sure. This happens when the algorithm is stable and all the particles are close together. Otherwise, the current frame will not be classified in order to avoid failing. The stability of the process will depend on the threshold selected as minimum goodness value for the particles.

A special situation that should be detected is unknown rooms. This happens when a frame corresponds to a new room not available during the training process. We propose studying the particle distribution after applying the odometry step. If most of these particles obtain a new

position beyond the limits of the original scenario, the robot will have entered an unknown room.

We assume the robot's movements for the test sequences will be similar to those obtained with the training sequences. The most common movement obtained from training sequences is going straight ahead. The algorithm will detect unknown rooms if the following situation arises:

- Instant t (before odometry step)
  - Most particles are in the same room $R1$
  - Positions $< x, y >$ of these particles are close to a room's bound
  - Particles are oriented to this bound
- Instant t (after odometry step)
  - Most particles are outside the room $R1$
  - Positions $< x, y >$ of these particles do not correspond to trained rooms

In this situation, the algorithm will classify the room as unknown. We selected 25% as the minimum percentage of particles outside a room to label the current frame as unknown.

5 Experiments and Results

The experiments were carried out using the KHL-IDOL2 [8] database as training and test sequences. Both tracks (obligatory and optional) can be evaluated using a script (provided by the competition organization) that obtains the score for a single test sequence. This script can obtain the score for six test sequences, available during the training period (2 sequences for each set of illumination conditions). For the final experiment, algorithms must be trained using a specific data sequence. Testing was performed on a test sequence with 1690 frames. This final test sequence couldn't be evaluated using the script, and the final results were obtained by submitting the classified test sequence to a dedicated web page.

For both tracks, we will show the results of applying our approach with different combinations of training and test frame sequences.

5.1 Obligatory track

For this track, it was only necessary to define the number of training frames to be used in order to classify a test frame. Using a huge values for this parameter, we accumulate noise from all the frames. We obtained unsatisfactory results due to the disparity between the number of training frames for the different rooms. For most training sequences, the number of corridor frames was always much bigger than the number for the other rooms. With a balanced training sequences this problem would disappear because the noise will be homogeneous for all the rooms. Using only the best training frame (instead of a number n of frames) we obtained bad results for the test frames belonging to the frontier of two different rooms.

Based on the experimental results, the best value for the number of training frames used to classify a frame was 20. All the experiments were performed using that value.

5.1.1 Preliminary Experiments

The experiments were carried out using the KHL-IDOL2 training and test sequences. We trained three systems using the different illumination conditions. Each one of these systems was tested using three frame sequences (obtained under different lighting conditions). Test and training sequences were different. We repeated the experiment for each combination of test and training data sequences. Tab. 3 shows the final score, the number of correctly (C) and misclassified (M) images, and the number of not classify (NC) images.

Training Seq.	Test Sequences											
	Night (952 frames)				Cloudy (928 frames)				Sunny (909 frames)			
	Score	C	M	NC	Score	C	M	NC	Score	C	M	NC
Night (1034 fr.)	**531**	643	224	85	285	433	265	230	265.5	421	311	177
Cloudy (915 fr.)	270.5	426	311	215	**404.5**	538	267	123	420.5	534	227	148
Sunny (950 fr.)	285.5	435	299	218	358.5	457	197	247	**509**	615	212	82

Table 3: Score, number of correct, incorrect and non classified frames for different combinations of training and tests frame sequences

The data shown in Tab. 3 demonstrates that the system developed highly depends on lighting changes. There is an important negative impact of using different illumination conditions for training and test sequences. Best results, for all the test sequences, were always obtained using the training sequence acquired under the same illumination conditions.

5.1.2 Final experiment

The final results obtained using the final image test sequence was published in the official track website. The final score obtained was 511.0. Complete results can be observed in Tab. 4.

Final Test Sequence			
Score	Correctly Classified	Misclassified	Not Classified
511.0	676	330	684

Table 4: Score, number of correct, incorrect and non classified frames for the best run in the obligatory track

The winner of this track was the Glasgow University (Multimedia Information Retrieval Group), which obtained 890.5 as final score. Our proposal obtained the 10th position (21 different runs were submitted). This result was not successful, but allows us to justify our proposal for the optional track.

5.2 Optional track

For the optional track, different parameters can be defined to select the best system configuration. The most important parameter is the number of particles (n). Another important parameter to define is the threshold to be used to define the algorithm stability, which should be tuned taking into account the expected variations for the lighting conditions. As mentioned in section 4.2.4 when the best particle weight is below this threshold the algorithm becomes unstable, and therefore a population initialization is performed.

For example, a value for this threshold of 7% of common points between two frames is a small value when given the same lighting conditions. Otherwise that percentage can be suitable if training lighting conditions are different from test lighting conditions.

For the final experiment, where training frames were taken under night illumination conditions, which are completely different from the test lighting conditions, the minimum threshold was set to 5%. This value was maintained for all the experiments, but using the same illumination conditions for training and test, this threshold value should be greater.

The final number of particles (n) will be 24 for all the experiments. Using this small number, a complete test sequence ($\approx$ 960 frames) can be classified in 56 minutes. The execution time for the algorithm is not fixed, and it will be higher for a big number of re-initializations.

5.2.1 Preliminary experiments

Some preliminary experiments were performed before the final test sequence was available. These experiments were carried out using the same training and test sequences than those used for the preliminary experiments performed in Section 5.1.1. Tab. 5 shows the final score, the number of correctly (C) and misclassified (M) images, and the number of not classify (NC) images.

		Test Sequences											
Training Seq		Night (952 frames)				Cloudy (928 frames)				Sunny (909 frames)			
		Score	C	M	NC	Score	C	M	NC	Score	C	M	NC
	Night (1034 fr.)	**837.5**	861	47	44	534.0	635	202	91	476.5	560	167	182
	Cloudy (915 fr.)	600.5	695	189	68	680.5	748	135	45	733.5	774	81	54
	Sunny (950 fr.)	725.0	791	132	29	**701.0**	769	136	23	**798.5**	823	49	37

Table 5: Score, number of correct, incorrect and non classified frames for different combinations of training and tests frame sequences

We can observe how the method obtains good results using training and test sequences acquired under different lighting conditions. For all the test sequences, (at least) 60% of frames were correctly classified. It is worthy of note that scores obtained for the different test sequences are not so strongly dependent on training illumination conditions as those obtained in Tab. 3. What is more, the best score for cloudy test frames was obtained with sunny training sequence. Making a comparison between these results and those obtained for the obligatory track, we can state that the localization method obtains significantly better results for the optional task.

5.2.2 Final experiment

Different runs of the algorithm were submitted to the website and the best score obtained was 916.5. This score was the highest for all submitted runs of all participants. In view of the obtained results, SIMD[4] group of University of Castilla-La Mancha was the winner for the optional track. Complete results can be observed in Tab. 6.

Final Test Sequence			
Score	Correctly Classified	Misclassified	Not Classified
916.5	1072	311	217

Table 6: Score, number of correct, incorrect and non classified frames for the best run in the optional track

Only three groups presented a proposal for this track, and the three best scores were 916.5, 884.5 and 853.0. Making a comparison of our proposals for the two tracks, it can be noticed that the final score was significantly improved (from 511.0 to 916.5). The number of misclassified frames was similar for both proposals, but the localization system used for the optional track increased the number of correctly classified frames. Because of this, the number of non classified images decreased.

6 Conclusions and Future Work

According to the results obtained for the optional track, our proposal becomes a robust alternative to traditional localization methods for indoor environments. It uses the principles of particle filter and Scale-Invariant Feature Transform to estimate the pose of the robot. The short execution time of our proposal allows the system to be used in real time. The system works properly with variable lighting conditions and changing indoor environments.

[4] *Sistemas Inteligentes y Minería de Datos*

The results obtained for the obligatory track shows the important problem of using SIFT with lighting changes. SIFT must be complemented with other techniques if lighting changes appear. The long execution time for feature extraction and matching makes it necessary to reduce the number of comparisons to be performed.

For future work, we aim to develop a general localization system capable of being trained automatically using the robot and its vision system. We shall also study the possible integration of the information extracted from distance sensors and odometry.

Acknowledgements

This work was partially supported by the Spanish "Junta de Comunidades de Castilla-La Mancha (Consejería de Educación y Ciencia)" under PCI08-0048-8577 and PBI-0210-7127 Projects, FEDER funds, and the Spanish research programme Consolider Ingenio 2010 MIPRCV (CSD2007-00018).

References

[1] B. Caputo, A. Pronobis, and P. Jensfelt. Overview of the CLEF 2009 robot vision track. In *CLEF working notes 2009*, Corfu, Greece, 2009.

[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE International Conference on Robotics*, 1999.

[3] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25(3-4):195–207, 1998.

[4] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11(391-427):27, 1999.

[5] V. Hough and C. Paul. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654.

[6] D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157. Corfu, Greece, 1999.

[7] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[8] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. The KTH-IDOL2 database. Technical Report CVAP304, Kungliga Tekniska Hoegskolan, CVAP/CAS, October 2006.

[9] Rudy R. Negenbornj. Kalman filters and robot localization. Master's thesis, Institute of Information and Computer Science, Utrecht University, Utrecht, Netherlands, 2003.

[10] Akira Nishino and Yasunari Shidama. The taylor expansions. *Formalized Mathematics*, 12 (2):195–200, 2004.

[11] Z. Wasik and A. Saffiotti. Robust color segmentation for the robocup domain. *Proc. of the Int. Conf. on Pattern Recognition (ICPR)*, 2:651–654, 2002.