

A comparison of algorithms for likelihood approximation in Bayesian localization

Andrea Censi

*Università degli Studi di Roma “La Sapienza”
Dipartimento di Informatica e Sistemistica “A. Ruberti”
via Eudossiana 18, I-00184 Rome, Italy
andrea.censi@dis.uniroma1.it*

Abstract—Global localization is the problem of estimating the robot’s pose, without any previous information, in a world which is known. Bayesian filters (either grid- or particle-based), need to evaluate the likelihood of a sensor reading given a pose hypothesis. For a laser range finder the computation is expensive and therefore approximations to the true likelihood that do not require a ray-tracing are usually employed. Nevertheless, these approximations do not satisfy those that in this paper we identify as *orientation* and *visibility* constraints. We propose more accurate algorithms which solve, in part or completely, these problems and are faster and have better complexity than existing methods.

I. INTRODUCTION

Global localization is the problem of estimating the robot’s pose, without any previous information, in a world which is known. It is both a necessary problem per se and it is a sub-problem of some approaches to Simultaneous Localization and Mapping (SLAM) where *closing the loop* means re-localizing the robot in the previously built map [1].

A central problem in global localization is handling uncertainty and Bayesian techniques have proven to be convenient. In that framework, the localization problem is formalized as to estimate $p(\mathbf{x}_t | \mathbf{u}_1, \mathbf{z}_1, \dots, \mathbf{u}_t, \mathbf{z}_t) := \text{Bel}(\mathbf{x}_t)$, that is the distribution of the current pose $\mathbf{x} = (x, y, \theta)$ given the history of odometry (\mathbf{u}) and sensor (\mathbf{z}) readings. Given some obvious independence assumptions, there is a recursive solution, which allows for designing a filter that needs to memorize only the current belief.

$$\text{Bel}(\mathbf{x}_t) \propto p(\mathbf{z}_t | \mathbf{x}_t) \int p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) \text{Bel}(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \quad (1)$$

The elegance in the theory is that (1) can be applied with different representation for the state space, obtaining grid-based [2] or particle-based filters [3]. Regardless of the representation, a Bayesian filter works in two steps: in the *prediction* step, the current state is predicted with the odometry measurement via the integral part of (1); in the *update* step, the distribution is weighted by the likelihood $p(\mathbf{z}_t | \mathbf{x}_t)$ to incorporate the sensor information.

Usually in the update step it is needed to evaluate point-wise $p(\mathbf{z}_t | \mathbf{x}_t)$ for a fixed observation $\bar{\mathbf{z}}_t$ and state $\bar{\mathbf{x}}_t$ (a particle or a cell). When using a laser range finder as a sensor, the computation can be computationally intensive. Given that the output of a range finder is a radial sampling $\mathbf{z}_t = \{(\phi_i, \rho_i)\}$,

assuming i.i.d. Gaussian noise, the likelihood can be modeled as

$$p(\mathbf{z}_t | x, y, \theta) = \prod_i \mathcal{N}(\rho_i - r_m(x, y, \theta + \phi_i), \sigma_l^2) \quad (2)$$

where $r_m(x, y, \psi)$ is the distance of the nearest obstacle from point x, y in direction ψ , according to the map m . Evaluating (2) exactly requires a ray-tracing operation on the map for each pose (particle or cell) and for each sensor point, therefore practical real-time implementations require either a huge data structure for precomputing such data or some approximations to the model [2], [4].

In particle filters, the knowledge of the likelihood is used also for *sensor resetting* [5], a technique which makes the filter more robust to modeling assumptions and to map uncertainty: every once in a while a number of particles is drawn directly from $p(\mathbf{x}_t | \mathbf{z}_t) \propto p(\mathbf{z}_t | \mathbf{x}_t)$, assuming a uniform prior. A similar procedure is also useful for a smart initialization of the filter: drawing the initial particles from an informed distribution $p(\mathbf{x}_0 | \mathbf{z}_0)$ reduces the number of particles needed.

The purpose of this paper is two-fold: we want to highlight some of the problems associated to the usual sensor model approximations which do not require a ray-tracing operation and propose novel algorithms which solve, in part or completely, these problems. All four algorithms presented here share the same philosophy: they do not need any assumption about the shape of the environment (curved, structured) or the presence of specific features (corners, lines); they are also very simple, easy to implement and analyze, and with a limited number of parameters.

The benchmark that we used for comparing the algorithms is the sensor resetting scenario, because it highlights the unnecessary ambiguities that arise when using an approximation of the likelihood which does not take into account the orientation and visibility constraints. Moreover assuming a uniform prior makes analysis easier because we will not be concerned with any particular evolution model and we will be able to employ a simple performance index, very robust to modeling assumptions.

In the following we reserve a section for each algorithm; a quantitative comparison is postponed until the last section.

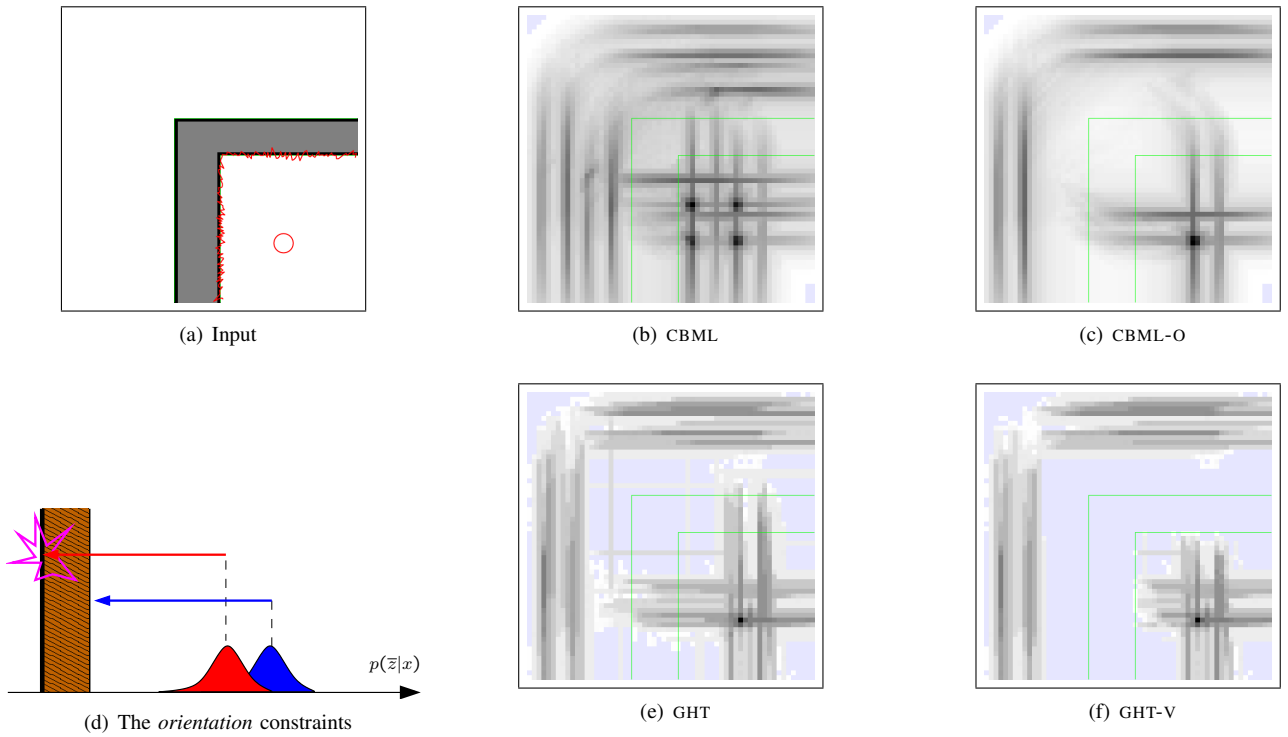


Fig. 1. Not taking into account the orientation of the surfaces creates false positives, for example in the typical situation of the double wall. A localization filter employing a likelihood approximation similar to CBML (Fig 1(b)) will never converge to the right solution, even though there is enough information in the data.

In this paper we collected the output of the algorithms in a three-dimensional grid. For visualization purposes, it is shown, for each pixel x, y , $\max_{\theta} p_i(\mathbf{z}|x, y, \theta)$. Output is linearly normalized, with black indicating high-likelihood areas; turquoise areas indicate a likelihood of exactly 0.

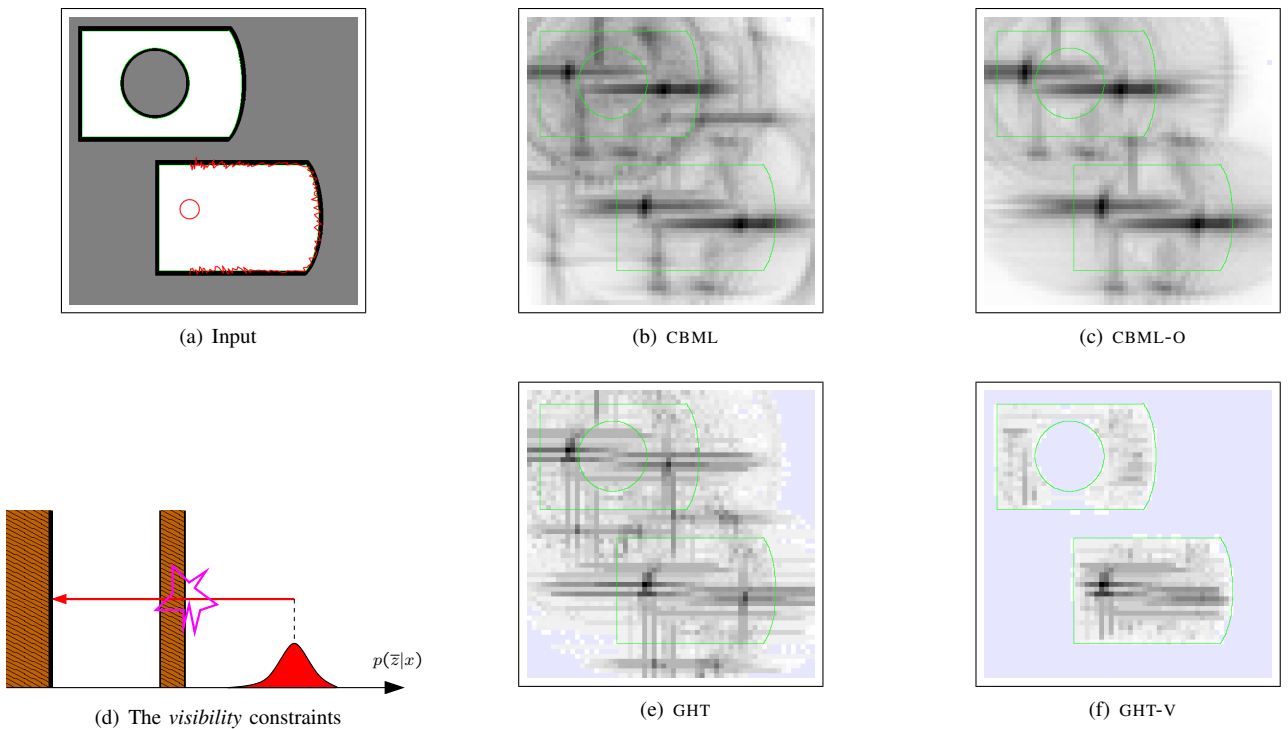


Fig. 2. Visibility constraints can be used to discriminate ambiguous situations. They are qualitatively different from the orientation constraints, because they involve spatial relationships between distant part of the map. Of the four algorithms described in this paper, only GHT-V is able to satisfy these constraints: as we can see from Fig. (f), it can resolve the ambiguity.

II. CBML

The main idea of Correlation Based Markov Localization (CBML) [6] is to use a 2D correlation operation for evaluating the similarity, and thus the likelihood, of the sensor scan, for each pose in the state space. We chose this algorithm from the literature because it is representative of the class of approximations that do not require ray-tracing nor a complex data structure.

The first, offline, step consists in computing an occupancy grid, by blurring the original map with a convolution kernel K . Then, in the actual localization step, the likelihood of each pose $\mathbf{x} = (x, y, \theta)$ is computed by correlating the scan, roto-translated at (x, y, θ) , with the occupancy grid map.

$$\mathcal{L}[x, y, \theta] = \int (\text{map} * K)[\mathbf{s}] \cdot (\text{scan} \oplus (x, y, \theta))[\mathbf{s}] ds \quad (3)$$

In practice (Algorithm 1) each sensor point is roto-translated by (x, y, θ) to find the corresponding point in the map; the value at the point in the occupancy grid is added to the likelihood.

Parameters: The convolution kernel K is chosen according to the sensor and world models¹ and the choice does not affect the run-time. For example, if one assumes to have a correct map, then the kernel can be chosen to be an isotropic Gaussian kernel with variance equal to the estimated sensor variance. Uncertainty in the map can be handled by increasing the kernel variance. As for the cell width, it makes sense to choose the same value for both the map grid and the results grid. This value should be coherent with the choice of the kernel, otherwise one would have over/under-sampling problems.

Qualitative analysis: We postpone a quantitative analysis of the results until Section VI. Here, we want to highlight two evident problems that deteriorate the quality of the results. Both derive from the simplification of the sensor model: the algorithm does not need ray-tracing, but sometimes the approx-

¹In [6] it was proposed to render also the sensor scan to an occupancy grid, and then correlate the scan occupancy grid with the map occupancy grid. We note that this is inefficient as it implies a rendering of the sensor scan for each direction. Our approximation is justified by the observation that it is possible to find a K such that $(a * K_m) \text{ corr } (b * K_s) \approx (a * K) \text{ corr } b$.

imation is poor. The first problem is illustrated in Fig. 1(d). As CBML does not take into account the surface orientation, there can be false positives. For example, in Fig. 1(a) there are parallel surfaces of different orientation; the CBML result in Fig. 1(b) presents three false positives. The second problem is that CBML ignores visibility constraints [6]. It follows that the output is not adequate in situations where the ambiguity could be resolved by considering these constraints, see for example Fig. 2. The impact on the localization procedure can be severe. If the robot remains in the same area, the estimate will never converge to the right solution, even if it obviously has enough information to localize itself. In particle filters it could happen that sample impoverishment leads to the wrong hypothesis being selected [7]. It is also possible to build example situations in which the wrong hypotheses have greater likelihood than the right ones: the estimate will converge to a wrong one (Fig. 4). The following algorithms have been designed to solve one or both of these problems.

III. CBML-O

We developed a variant of CBML which takes into account the orientation information in the data, interpreting both the sensor and the map as vector fields.

The first step is to estimate the surface orientation for each sensor and map point (Fig. 3). We will not go in details about this step; we used regression for the sensor and pattern matching for the map. Then we use those information to create a vector field $\vec{\text{map}}$ that, after blurring, will be used as our map (Fig. 4).

The correlation operation is therefore interpreted as a scalar product between vector fields.

$$\mathcal{L}[x, y, \theta] = \int (\vec{\text{map}} * K)[\mathbf{s}] \cdot (\vec{\text{scan}} \oplus (x, y, \theta))[\mathbf{s}] ds \quad (4)$$

In practice (Algorithm 3) each oriented point in the scan is roto-translated by (x, y, θ) , then a scalar product is computed with the corresponding point in the map. If the two vector fields have a compatible direction at that point (difference is $< 90^\circ$), the scalar product is positive and gives a positive contribution to the correlation; this enforces the orientation constraints (Fig. 1(c)).

Algorithm 1 - CBML algorithm

```

1: {For each pose}
2: for all  $(x, y, \theta)$  do
3:   {For each sensor point}
4:   for all  $\mathbf{p}_i$  do
5:      $\mathcal{L}[x, y, \theta] += \text{bmap}[\mathbf{p}_i \oplus (x, y, \theta)]$ 
6:   end for
7: end for

```

Algorithm 2 CBML algorithm - faster

```

1: for all  $\theta$  do
2:   for all  $\mathbf{p}_i$  do
3:      $\mathbf{p}' = \mathbf{p}_i \oplus (0, 0, \theta)$ 
4:     for all  $(x, y)$  do
5:        $\mathcal{L}[x, y, \theta] += \text{bmap}[\mathbf{p}' \oplus (x, y, 0)]$ 
6:     end for
7:   end for
8: end for

```

In these pseudo-code listings, $\oplus(x, y, \theta)$ means a rotation of θ followed by a translation (x, y) . We do not show the buffer's normalization.

$\text{bmap} = \text{map} * K$ is the blurred map used by CBML- see Fig. 4. It has been claimed that CBML is easily optimizable with processor-specific SIMD operations; this can be easily shown by reordering the *for* loops in Algorithm 1, where the inner loop performs a costly roto-translation. Algorithm 2 is equivalent but its inner loop is a simple filter that can be easily optimized.

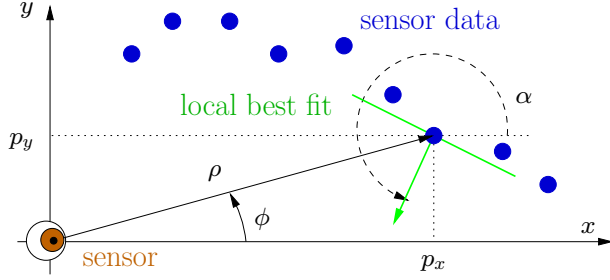


Fig. 3. The output of a range finder is usually a list of readings in polar coordinates $\mathbf{z}_t = \{(\phi_i, \rho_i)\}$. Throughout the paper we refer to the Cartesian representation: $\mathbf{p}_i = (\rho_i \cos \phi_i \ \rho_i \sin \phi_i)^T$. Moreover, we associate to each sensor point an angle α_i which is an estimate of the surface orientation at that point obtained using linear regression.

IV. GHT

We consider a direct application of the Generic Hough Transform (GHT) [8] as a likelihood approximation algorithm. We already used the GHT, a tool from computer vision, for solving the scan matching problem [9].

Also the GHT needs a representation of both the map and the sensor scan as a list of oriented points: let the two sets be $\{(\mathbf{p}_i, \alpha_i)\}$ and $\{(\mathbf{p}_j, \alpha_j)\}$. Suppose that we knew that a point (\mathbf{p}_i, α_i) in a set corresponds to a point (\mathbf{p}_j, α_j) in the other set with a rotation θ and a translation \mathbf{t} :

$$\mathbf{p}_j = R_\theta \mathbf{p}_i + \mathbf{t} \quad (5)$$

$$\alpha_j = \alpha_i + \theta \quad (6)$$

In this expression, R_θ is a 2×2 rotation matrix. These relations can be inverted to find θ and \mathbf{t} :

$$\hat{\theta} = \alpha_j - \alpha_i \quad (7)$$

$$\hat{\mathbf{t}} = \mathbf{p}_j - R_{\hat{\theta}} \mathbf{p}_i \quad (8)$$

$\hat{\mathbf{x}} = (\hat{\mathbf{t}}, \hat{\theta})$ is therefore the robot pose that makes \mathbf{p}_i and \mathbf{p}_j match. The GHT applies (7) and (8) repeatedly for each pair of a sensor point and a map point; each pair generates an hypothesis. Each hypothesis can be considered a particle by itself [9]; in this paper we accumulate the result in a grid for easy comparison with the other algorithms (Algorithm 4).

Parameters: GHT is almost parameter-less, the only degree of freedom being the cell width for the result grid. We found experimentally that the optimal value of the cell size that

trades between over/under-sampling is consistently a little less than the deviation of the sensor noise - although it varies also with other factors, such as the shape of the environment. The run time of the algorithm does not change if the cell width is different (GHT has very interesting complexity properties: see the appendix). Also note that GHT needs to memorize only the surface points in the map, allowing for a more compact representation. *Qualitative analysis:* The matching between pairs of points performed by GHT assures that the surfaces' orientation be consistent (Fig. 1(e)). As for the visibility constraints, they are ignored (Fig. 2(e)).

V. GHT-V

We propose a light modification of GHT that takes into account the visibility constraints, by employing a *visibility map* that represents the visibility relations between the surface points. To each point surface \mathbf{p}_j in the map, we associate an array $\text{vis}_j[\psi]$ containing, for every angle ψ , the distance to the nearest obstacle in that direction. This representation is much more compact than precomputing the whole sensor model, because it interests only the surface points.

Once we have this new data structure, we just need to add to GHT a test that checks whether the point \mathbf{t} is visible from the surface point \mathbf{p}_j (Algorithm 5); this enforces the satisfaction of the visibility constraints (Fig. 2(f)).

Parameters: GHT-V has another parameter: the angle resolution for the ray-tracing visibility array. It is useless to choose it greater than the resolution for θ ; in practice we had good results with very coarse discretization.

Note that GHT and GHT-V do not enumerate the poses explicitly; this is not a problem for grid-based localization, but in a particle filter one wants to evaluate the likelihood for single particles, so it would be needed to memorize the GHT results in an intermediate grid. Even though GHT is quite fast (see the experiments section), this could preclude its use at each time step and suggests to employ it every once in a while to cut hypotheses based on the visibility constraints.

VI. EXPERIMENTS

The expression of the likelihood in (2) does not take into account the uncertainty in the map. Small variations of the map can create big variations to the term $r_m(x, y, \psi)$ in (2); for a

$$2 \frac{\partial r_m(x, y, \psi)}{\partial m} \text{ is not bounded.}$$

Algorithm 3 CBML-O algorithm

```

1: for all  $(x, y, \theta)$  do
2:   for all  $(\mathbf{p}_i, \alpha_i)$  do
3:      $v_1 = \overrightarrow{\text{vmap}}[\mathbf{p}_i \oplus (x, y, \theta)]$ 
4:      $v_2 = (\cos(\alpha_i + \theta) \ \sin(\alpha_i + \theta))^T$ 
5:      $\mathcal{L}[x, y, \theta] += v_1^T v_2$ 
6:   end for
7: end for

```

Algorithm 4 GHT algorithm

```

1: for all  $(\mathbf{p}_j, \alpha_j)$  do
2:   for all  $(\mathbf{p}_i, \alpha_i)$  do
3:      $\theta = \alpha_j - \alpha_i$ 
4:      $\mathbf{t} = \mathbf{p}_j - R_\theta \mathbf{p}_i$ 
5:      $\mathcal{L}[\mathbf{t}.x, \mathbf{t}.y, \theta] += 1$ 
6:   end for
7: end for

```

$\overrightarrow{\text{vmap}} = \overrightarrow{\text{map}} * K$ is the blurred vector field used as a map by CBML-O (Fig. 4).

single reading, the real likelihood, subject to map uncertainty, is multi-modal and cannot be well approximated by a Gaussian [6]. For this reason, instead of directly comparing (2) to the output of the algorithms, we devised a performance index which does not need the point-wise evaluation of the likelihood and is therefore more robust to our modeling assumptions.

We employ the Kullback-Leibler Divergence (KLD)³ for evaluating the similarity between the approximated and the target distribution, considering p_i as a better approximation than p_j to the target distribution p_0 if

$$\text{KLD}(p_0, p_i) < \text{KLD}(p_0, p_j) \quad (10)$$

Simple manipulation of (10) leads to the following equivalent expression:

$$E_{p_0}[\log p_i(\mathbf{z}|\mathbf{x})] > E_{p_0}[\log p_j(\mathbf{z}|\mathbf{x})] \quad (11)$$

We compute this by Monte Carlo integration. We sample from $p_0(\mathbf{x}, \mathbf{z})$, by drawing a pose $\mathbf{x}^{(k)}$ from the prior $p_0(\mathbf{x})$, assumed uniform, then simulating a sensor scan $\mathbf{z}^{(k)}$ according to $p_0(\mathbf{z}|\mathbf{x}^{(k)})$. The final expression for the performance index is

$$S_i = \frac{1}{n} \sum_{k=1}^n \log p_i(\mathbf{z}^{(k)}|\mathbf{x}^{(k)}) + \log V \quad (12)$$

The offset $\log V$, where V is the volume of the \mathbf{x} space, allows for comparing experiments with environments of different size⁴. Experimentally we found that (12) tends to converge after a relatively small number of iterations (100).

³The KLD is defined for two distributions p_a, p_b as

$$\text{KLD}(p_a, p_b) = \int p_a(\xi) \log \frac{p_a(\xi)}{p_b(\xi)} d\xi \quad (9)$$

Important properties are: $\text{KLD}(p_a, p_b) \leq 0$, and $\text{KLD}(p_a, p_b) = 0 \Leftrightarrow p_a = p_b$; note however that $\text{KLD}(p_a, p_b) \neq \text{KLD}(p_b, p_a)$ therefore it is not a "distance" in the space of distributions.

⁴Consider two experiments of the i -th algorithm in two environments E_A and E_B , which is composed by two exact copies of E_A juxtaposed. In this case we expect the performance index to be the same in the two experiments. But $p_i^B(\mathbf{z}^{(k)}|\mathbf{x}^{(k)}) = \frac{1}{2} p_i^A(\mathbf{z}^{(k)}|\mathbf{x}^{(k)})$, and we obtain that $E_{p_0}[\log p_i^B(\mathbf{z}|\mathbf{x})] = E_{p_0}[\log p_i^A(\mathbf{z}|\mathbf{x})] - \log 2$, therefore we add a compensating factor for volume.

Experimental results: The simulated sensor that we employed has these specs: 181 rays, 180° field of view, Gaussian noise with $\sigma = 2cm$. Cell size was 10cm, 30° for GHT, GHT-V, and 10cm, 3° for CBML, CBML-O, which require a smaller discretization.

The results are summarized in the following table (100 iterations). We found that the performance index is quite similar across different environments and that the qualitative analysis is confirmed: algorithms which satisfy the orientation and the visibility constraints produce an estimate closer the true distribution.

	Fig. 2	Fig. 5	curved env.
CBML	-0.47	-0.27	-0.35
CBML-O	0.71	0.65	0.96
GHT	2.85	2.10	2.20
GHT-V	3.71	2.93	2.91

Run-time: CBML-O has virtually the same runtime as CBML; likewise for GHT-V and GHT. Regarding CBML vs. GHT, the comparison is hard because they have different complexity, as illustrated in the appendix. [6] measured the run-time of CBML in seconds per pose per reading, obtaining a value of less than $10^{-9}s$ (our non optimized implementation runs at $7 \cdot 10^{-7}s$); this metric is not convenient for GHT as the algorithm does not enumerate the pose explicitly. For each pair of points an unoptimized implementation of GHT takes $1.5 \cdot 10^{-6}s$ on a PowerPC 1.5Ghz; using standard optimization techniques like fixed point math and look-up tables cuts the time 4x. For typical cases we found that GHT (not optimized) would run at about the same speed as the optimized version of CBML.

VII. CONCLUSIONS AND FUTURE WORK

While it is clear that our new algorithms approximate the target distribution better and have a better complexity, we think that each considered algorithm has its advantages: CBML is a relatively simple algorithm, but its approximation to the likelihood may cause problems and the unoptimized version is too slow for practical use; CBML-O has the same run-time and the same optimization possibilities as CBML but provides more accurate results; GHT has better results, and better complexity,

Algorithm 5 GHT-V algorithm

```

1: for all ( $\mathbf{p}_j, \alpha_j, \text{vis}_j$ ) do
2:   for all ( $\mathbf{p}_i, \alpha_i$ ) do
3:      $\theta = \alpha_j - \alpha_i$ 
4:      $\mathbf{t} = \mathbf{p}_j - R_\theta \mathbf{p}_i$ 
5:     if  $\text{vis}_j[\varphi(\mathbf{t} - \mathbf{p}_j)] > |\mathbf{p}_i|$  then
6:        $\mathcal{L}[\mathbf{t}.x, \mathbf{t}.y, \theta] += 1$ 
7:     end if
8:   end for
9: end for

```

Algorithm 6 Computing the visibility map

```

1: for all  $\mathbf{p}_i$  do
2:   for all  $\mathbf{p}_j \in \text{Ball}_\rho(\mathbf{p}_i)$  do
3:      $d = \|\mathbf{p}_j - \mathbf{p}_i\|$ 
4:      $\psi = \varphi(\mathbf{p}_j - \mathbf{p}_i)$ 
5:      $\text{vis}_j[\psi] = \min(d, \text{vis}_j[\psi])$ 
6:   end for
7: end for

```

The variable $\text{vis}_j[\psi]$ is the distance from p_j to the nearest obstacle in direction ψ ; it is a discretized version of $r_m(\mathbf{p}_j.x, \mathbf{p}_j.y, \psi)$. $\varphi(v) := \text{atan2}(v.y, v.x)$ Algorithm 6 is a way to compute the visibility map. By considering an horizon ρ for the sensor, it runs in linear time with respect to the area of the environment. Choosing a ρ less than the true horizon does not have a catastrophic effect on the output, it just lets a part of the visibility constraints to be ignored.

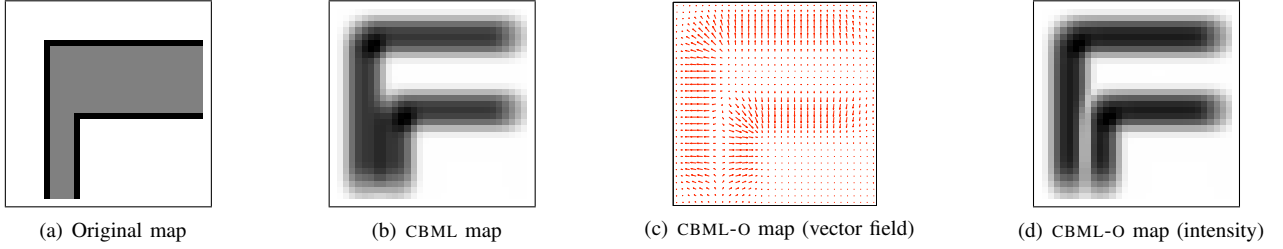


Fig. 4. In CBML, the surface points of the original map (Fig. a) are blurred as to produce an occupancy grid "bmap" (Fig. b) In CBML-O, the map is a blurred vector field (Fig. c): at each pixel it is associated a direction and an intensity. The vector field is created from the orientation estimate for each surface pixel as: $\vec{\text{map}}(x, y) = \sum_j (\cos \alpha_j \quad \sin \alpha_j)^T \delta_{\mathbf{p}_j}(x, y)$ and then a blurring operation is applied. CBML tends to merge walls that are close to each other, generating false maximum in the likelihood. This does not happen in CBML-O, because the two walls generate opposite fields.

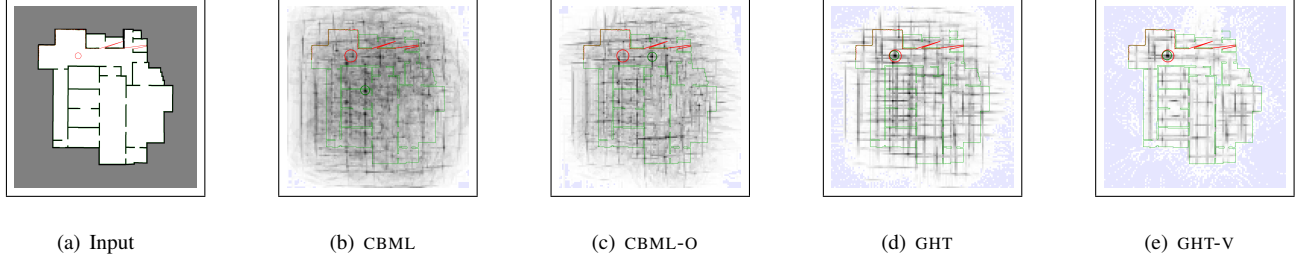


Fig. 5. Example run in a large environment. Bad results of CBML and CBML-O are due to two effects: double walls at the center of the environment tend to confuse the methods and the effects of the discretization of θ can be catastrophic for distant readings. Both CBML and CBML-O do an extensive search in a discretized state space. For example if the cell size for θ is set to 30° , the expected difference of orientation between a sampled pose and the real pose is 15° ; for a reading at $10m$, this means an error of over $2.5m$ which is enough to make the correlation inconclusive.

though it requires a sensor precise enough to use orientation information; GHT-V is the only one that respects the visibility constraints.

There is still space for improvement for GHT-V. In fact, interestingly enough, we found that the true pose is the absolute maximum of the estimated distribution only in approximately 70% of the trials (for CBML-O, it is only 47%). We think that this could be due to not keeping track of multiple votes by the same point in the same grid cell in Algorithm 5. Moreover, an accurate modeling of the uncertainty in (8) could lead to better results.

APPENDIX - COMPLEXITY COMPARISON

We show in this table the complexity of the algorithms referred to the sensor resetting scenario: computing the likelihood of a sensor scan with n readings in an environment of area A . Let ρ_m the x, y map resolution, and ρ_θ the angle resolution for the solution grid.

	init. time	storage	run time
GHT-V	$\mathcal{O}(A \cdot \rho_m^2)$	$\mathcal{O}(A \cdot \rho_m \cdot \rho_\theta)$	$\mathcal{O}(A \cdot \rho_m \cdot n)$
GHT	0	$\mathcal{O}(A \cdot \rho_m)$	$\mathcal{O}(A \cdot \rho_m \cdot n)$
CBML-O	$\mathcal{O}(A \cdot \rho_m^4)$	$\mathcal{O}(A \cdot \rho_m^2)$	$\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta \cdot n)$
CBML	$\mathcal{O}(A \cdot \rho_m^4)$	$\mathcal{O}(A \cdot \rho_m^2)$	$\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta \cdot n)$
Table	$\mathcal{O}(A \cdot \rho_m^3 \cdot \rho_\theta)$	$\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta)$	$\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta \cdot n)$

There are $\mathcal{O}(A \cdot \rho_m^2)$ pixels in the map, of which $\mathcal{O}(A \cdot \rho_m)$ are surface points, and there are $\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta)$ poses in the state space. Precomputing a ray-tracing table costs $\mathcal{O}(A \cdot \rho_m^3 \cdot \rho_\theta)$ and needs $\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta)$ storage. **CBML**: The storage required for memorizing the blurred map is $\mathcal{O}(A \cdot \rho_m^2)$. As for the time complexity, the correlation is a loop over the possible poses and the number of sensor points. Therefore the complexity is $\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta \cdot n)$. Note that because it does an extensive search of the state space, CBML has the same cost of using (2) with precomputed ray-tracing data - which, nevertheless, requires more storage. **GHT**: The complexity of GHT is $\mathcal{O}(A \cdot \rho_m \cdot n)$, less than the cost of an extensive search. **GHT-V**: Compared to GHT, memorizing the visibility map raises the storage requirements from

$\mathcal{O}(A \cdot \rho_m)$ (memorizing only the surface points) to $\mathcal{O}(A \cdot \rho_m \cdot \rho_\theta)$ (memorizing also a buffer for each surface point). Note however that the cost is less than memorizing the precomputed ray-tracing for all poses, which is $\mathcal{O}(A \cdot \rho_m^2 \cdot \rho_\theta)$. The runtime is the same as GHT.

As for the runtime, note that all the algorithms are linear in n and A , but the GHT algorithms do not depend on ρ_θ and have a complexity linear in ρ_m instead of quadratic as CBML and CBML-O. As for the storage requirements, GHT and GHT-V are linear in ρ_m , but GHT-V also depends on ρ_θ .

REFERENCES

- [1] J. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2000.
- [2] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," vol. 25, 1998, pp. 195–207.
- [3] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proc. of the National Conference on Artificial Intelligence*, 1999.
- [4] W. Burgard, D. Fox, and D. Hennig, "Fast grid-based position tracking for mobile robots," in *Proc. of the 21th German Conference on Artificial Intelligence, Germany*, 1997.
- [5] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA'00)*.
- [6] K. Konolige and K. Chou, "Markov localization using correlation," in *Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI'99)*, 1999.
- [7] A. Milstein, J. Sanchez, and W. E.T., "Robust global localization using clustered particle filtering," in *Proceedings of the 18th National Conference on Artificial Intelligence - AAAI - Edmonton, Canada*, 2002.
- [8] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," in *Readings in computer vision: issues, problems, principles, and paradigms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987, pp. 714–725.
- [9] A. Censi, "Scan matching in a probabilistic framework," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA'06)*, 2006, to appear. [Online]. Available: <http://www.dis.uniroma1.it/~censi/>