# Conformative Filter : A Probabilistic Framework for Localization in Reduced Space

Chatavut Viriyasuthee and Gregory Dudek
Centre for Intelligent Machines
McGill University
Montreal, Quebec, Canada H3A 2A7
{pvirie,dudek}@cim.mcgill.ca

*Abstract*— **Algorithmic problem reduction is a fundamental approach to problem solving in many fields, including robotics. To solve a problem using this scheme, we must reduce the problem into another one for which solutions exist. The reduction function, which infers a conformation between the problem and the solution space, plays an important role in solution evaluation and is sometimes used to transform the solutions into the problem domain.**

**We consider robot path planning in the context of algorithmic problem reduction where a reduction can be used to adapt a path (referred to as solution) generated by a human or other subsystem to environmental constraints that may differ from those at plan-generation time. Usually, solving these problems involves estimating the current state in the plan and trying to retrieve the solution. We develop a probabilistic framework for reduction-based path planning where the solutions can be obtained from localization into the plan by exploiting the Markov property. We name it *Conformative Filter*. The algorithm is an extension of Bayes' filter which tries to search for not only the solutions but also conformation between the environment and the plan. An implementation based on Localization and Expectation-maximization is discussed along with evaluation on navigation tasks using a set of actual hand-drawn maps of simulated environments. The results demonstrate applicability and effectiveness of the algorithm in such tasks and show that the proposed filter results in improved localization when compared with conventional approaches.**

## I. Introduction

We are interested in path planning methods (and related techniques) where preliminary paths (so-called solutions) generated by some subsystems are executed in environments that are different from that known when the planning was performed. In particular, we are interested using plans generated by humans on schematic maps in the form of hand-drawn sketches to control agents in real environments. Transforming schematic plans to real control actions involves 1) comparing the observed features in the environments to the plans, 2) identifying the current steps, and 3) retrieving the actions. These three processes strongly resemble to the processes required to execute a single navigation loop in traditional robotic systems. In many cases and, in particular when humans use maps for planning, the plan and environment representation are necessarily highly abstracted and the plan is usually acquired through a *reduction* process that leaves some details from the environment abstract, inaccurate, or completely absent; thus the localization process and plan execution process very difficult. Many approaches have been proposed for navigation using potentially inaccurate maps, but they are usually limited by domain-specific assumptions which we wish to avoid. For example, Tomono et al. present an effective navigation approach in hybrid topological-metric maps [1]. They assume that the maps consist of many local islands of accuracy and connected by global relations corrupted by Gaussian noise between them. Setalaphruk et al. adopt topological maps to guide agents in limited environments such as corridor and hallway using a basic obstacle avoidance control [2]. Skubic et al. pose their idea on navigation using hand-drawn maps in object spaces [3]. Their approach is literally a version of reduction where they parse the environments into navigation states.

By reduction, we refer to the process of converting an instance of problem into one another. It can also be used to retrieve a solution for the original problem by backward converting a solution from a similar problem whose the solution already exists or can be obtained effortlessly. Many ideas have been proposed under this scheme. Veloso presents a complete framework for storing and reusing past experience in newly-faced problems [4]. Plan reuse, repair, and adaption have been explored many times in previous decades [5], [6], though they were mostly limited to specific domains. Planning by reduction has a strong relation to case-based planning, where solutions to problems are found by exhaustive search from memory [7]. It is also related to using continuation methods akin to graduated non-convexity [8] to plan the solution in a simple domain and graduately transform the solution into the problem domain. Furthermore, some external constraints may be incorporated into the search procedure to identify good solutions as presented in [9]. Moreover, this idea can be regarded as learning by imitation [10], and it is also closely tied to the tracking problem regardless of retriving solutions. Some of sophisticated approaches in these areas are [11]–[13]. The most recent work that captures this concept has been deployed in multi-robot control [14]; it is inspired from the concept of registration in computer vision community. Notably, the idea of using computer vision mehods to interpret hand-drawn sketch maps and related them to the real world goes back to the classic work by Mackworth [15].

In this paper, we present a reduction-based algorithm for path planning in environments with the state transitions that satisfy the Markov property. The idea can also be expanded into other domains such as navigation using an inaccurate map,

978-0-7695-4362-8/11 $26.00 © 2011 IEEE
DOI 10.1109/CRV.2011.11

learning by demonstration, or adaptive control. Unlike existing work, we use a probabilistic model to address the search problem and derive an algorithm to evaluate the conformation from the cognitive map [16]. The algorithm is based on Bayes' filter and it searches for solutions to the problems while trying to maximize the conformation between the environments and the plans following the concepts of Localization and Expectation-maximization (EM) [17]. We test the algorithm on the problem of navigation using a hand-drawn map. The results along with discussions are provided.

## II. Localization in Reduced Space

In this section, we first discuss the characteristics of the inaccurate maps that we refer to as *Reduced Spaces*. To ground our work we consider a topological maps based on the cognitive map representation developed by Yeap et al.[16], since we seek to facilitate interaction with human users. Yeap presents a computational model of a cognitive map with a Relative Absolute (R-A) Model, where it contains a vague global representation of the world (referred to as Relative Space Representation, or RSR) that links together many small detailed-rich patches of local areas (called Absolute Space Representation, or ASR). To further limit the scope of our work, we are only concerned with continuous environments, where the entire space can be presented by an ASR. The ASR may be created from past experience through egocentric perception, which renders it suffer from metric distortion and missing details [16]. Creating the ASR by this scheme, the reduction function from the world to the ASR must be defined by the creator. This guarantees many-one reduction from the world to the ASR, but it is only limited to local area because the ASR may be created from concatenation of perception sequences. We set the definition of our solution space to be an ASR where local many-one reductions from the world to the ASR are permitted. We refer to this ASR as *reduced space*.

The reduced space represents a map or a plan where the solutions exist or are easy to obtain. To solve our navigation problem, we must localize the agent into the reduced space. The localization results may be one-to-many, because some features in the environment may redundantly be presented in different areas of the reduced space. The transitions between agent's states are required to limit the search scope. The advancement of Markov applications suggests ways to model the navigation sequence using the Markov property. However, the absence of the reduction function also adds difficulty to the localization. If the reduction function is known, the problem can be solved by Markov localization [18], but this is not always the case. Consequently, we wish to simultaneously find the reduction function while locating the conformed solutions given that function.

We split the function into two parts: the *explicit* part which can be directly inferred from the given cues and the *implicit* part that can be varied. Because the ease of learning the two parts are not equivalent, it is more convenient to learn them separately. For example, in image registration problems, the explicit part is color mapping and the implicit part is pixel location mapping. Color mapping may contain some errors but it is generally reliable enough for us to match them to the pixel colors in the template image (reduced space) and infer the position mapping in Expectation-maximization fashion. EM and Matching (same concept as EM-ICP) have been well studied in computer vision [19], [20]. The sources of model error are location and detection. Another type of error originates from the difference in assumption of the ideal function from the learned function. Mainly, we want to find the reduction function that minimizes these errors using the available cues.

We define the navigation problem using an inaccurate map as the process of localization in a reduced space. During the process, the reduction function has to be learned via the optimization routine that minimizes the conformation errors. This function and the state transitions are used to evaluate locations in the reduced space to find a suitable solution for the current situation.

## III. Conformative Filter

In this section, we present the *Conformative Filter* for localization in reduced space as an approach to select actions for inaccurate map navigation problems that satisfy the Markov assumption. The algorithm is an extension of Bayes' filter for a Hidden Markov Model where it is an augmented version of the conventional one with two state variables that present the explicit and the implicit part of the reduction function. Fundamental localization rules have been discussed along with supplementary rules based on the idea of Expectation-maximization.

### A. The model

In our model, each environment space and each reduced space are denoted by $W = W_v \times W_u$ and $M = R_v \times R_u \times Q$ respectively where $W_v$, $W_u$, $R_v$, $R_u$ are the set of features in the environment and the reduced space that correspond to the explicit part $v$ and the implicit part $u$ of the reduction function, $Q$ represents the space of solutions that we seek. The state variables are the following:

- $m \in M$ the location in the reduced space.
- $v : W \to R_v$ the explicit part of the function.
- $u : W \to R_u$ the implicit part of the function.
- $s \in R_v \times W_u$ the observation for uncovering $v$.
- $w \in W$ the observation in the environment.
- $r \in R_v \times R_u$ the expected observation in the reduced space.

In each navigation step, the possibilities of $m, v, u$ may propagate from the previous $m, v, u$ by the adopted solution given by the previous $m$. The choice of the current $m$ may also affect the current $v, u$ in the form of prior assumptions, where we can limit the set of possible $v, u$ at some locations in the reduced space that store specific shapes of the reduction function. $v$ and $u$ together compose the reduction function that transforms features in the environment to features in the reduced space. $s$ and $w$ are distinct observations that contain features from different domains, but $s$ cannot be used without $w$ because $v$

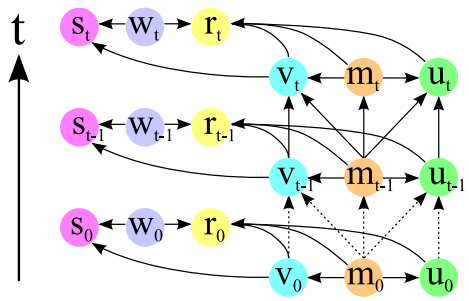



Fig. 1. The state model of the Conformative Filter for a navigation sequence. $m, v, u$ are the hidden variables, $s, w, r$ are the observations, and $t$ is the step index.

requires $w$ to create comparisons for $s$. The same concept is applied to $r$, where it requires either $m$ or $v, u, w$ to complete the observation. $r$ does not observe anything by itself, but its presence is necessary to interconnect $m, v, u, w$. Figure 1 and Figure 2 show the state model of the Conformative Filter and an example of the relations between the state variables respectively.

### B. Propagate and Update

Like the conventional recursive filters [18], [21], we intend to recursively learn $m, v, u$ from all available observations and the constraints posed by prior assumptions. The learning process can be decomposed into two fundamental rules: 1) *Propagate* $m, v, u$ using the previous adopted solution and prior assumptions. 2) *Update* $m, v, u$ using the new observations $s, w$. Let us define $Z_t = \{(s_t, w_t), (s_{t-1}, w_{t-1}), ..., (s_0, w_0)\}$: the collection of all observations up to a step $t \in \mathbb{N}$, and let state vectors with no subscription and prime notation belong to the current and the previous step respectively. According to the model in Figure 1, the propagate rule is given by the following:

$$P(v, u, m|Z') = \iiint P(v, u, m|v', u', m')P(v', u', m'|Z')\,\mathrm{d}v'\mathrm{d}u'\mathrm{d}m' \quad (1)$$

Where

$$\begin{aligned} &P(v, u, m|v', u', m') \\ &\quad = P(v|u, m, v', u', m')P(u|m, v', u', m')P(m|v', u', m') \\ &\quad = P(v|m, v', m')P(u|m, u', m')P(m|m') \end{aligned} \quad (2)$$

The update rule can be derived as below:

$$\begin{aligned} &P(v, u, m|Z) \\ &\quad = \int P(r, v, u, m|Z)\,\mathrm{d}r \\ &\quad = \int \frac{P(s, r|v, u, m, w, Z')P(v, u, m|w, Z')}{P(s|w, Z')}\,\mathrm{d}r \\ &\quad = \frac{P(v, u, m|Z')P(s|v, w)}{P(s|w, Z')} \int P(r|v, u, m, w)\,\mathrm{d}r \end{aligned} \quad (3)$$

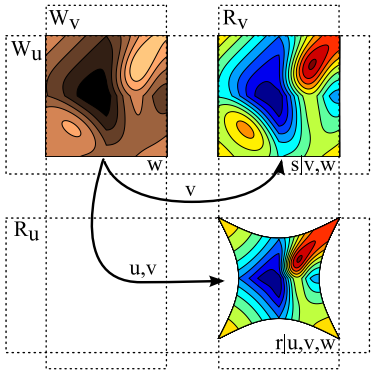



Fig. 2. An example of the relation between the state variables. $u$ serves as a location transform function, while $v$ is the collection of detected features in this space. The environment observation $w$ can be transformed to $s$ by $v$ and to the reduced space instance $r$ by $u$ and $v$.

for the first step:

$$P(v_0, u_0, m_0|Z_0) = \frac{P(v_0, u_0, m_0)P(s_0|v_0, w_0)}{P(s_0|w_0)} \int P(r|v_0, u_0, m_0, w_0)\,\mathrm{d}r$$

Where

$$P(v_0, u_0, m_0) = P(v_0|m_0)P(u_0|m_0)P(m_0)$$

The maxima of these probabilistic rules directly infer $m, v, u$ for each step. The integral term in Equation (3) (replaced by a summation in the case of discrete variables) presents the necessity to consider all possibilities of $r$. However, in practice, we can straightforwardly infer $r$ from $m$ or $u, v, w$ if either of them is known. We assume that $m_0$ is given, and $v_0$ and $u_0$ are predominantly influenced by the prior from $m_0$ because there is no previous step to draw propagation.

### C. Conformation

In the model, $m$ and $r$ correspond to the true and the observed state in the original Bayes' filter. Ideally, we want to learn $m$ from $r$, but the situation is complicated because $v, u$ can vary to create $r$ from $w$ that always satisfies $m$. Thus, $s$ is necessary as the additional cue to restrain $v$, which in turn forces the algorithm to choose the most likely $v, u$ and $m$. However, because there is always some uncertainty in $s$, $v$ has to be refined at each time we get a new cue. The main idea is analogous to an Expectation-maximization (EM) problem [17]. [1] Therefore, we derived the new update rules from Equation (3) following the E step and the M step in Expectation-maximization.

[1]When compared to EM, $v$ and $u$ are equivalent to the latent variables, $w \in W$ is the observed variable, and $m$ is the unknown parameter that defines the joint distribution $P(w, v, u|m)$. On the other hand, we can also treat $u$ as another unknown parameter, and try to optimize both $u$ and $m$ that maximize the joint distribution $P(w, v|u, m)$.

E-Update:

$$P(v|u,m,Z) = \alpha P(v,u,m|Z')P(s|v,w)\int P(r|v,u,m,w)\,\mathrm{d}r \quad (4)$$

M-Update:

$$P(u,m|v,Z) = \beta P(v,u,m|Z')\int P(r|v,u,m,w)\,\mathrm{d}r \quad (5)$$

E-Update optimizes $v$ to satisfy both $s, r$ and its propagated prior given $u, m, w$, while the M-Update optimizes $u$ and $m$ to satisfy $r$ and their priors given $v, w$. $\alpha, \beta$ are normalization terms that do not involve in the optimization processes. These two rules together form the update rule for $v, u, m$. They have to be alternatively iterated until the convergence is reached. A new question rises asking for the initial condition. Practically, we can ignore the integral term and use $s, w$ as the initial cues (since they are the only reliable cues now) to learn a good start of $v$ as follows:

$$P(v|u,m,Z) \approx \alpha P(v,u,m|Z')P(s|v,w) \quad (6)$$

Furthermore, when solving for the reduction function, the process may become underdetermined at any step, namely when the degree of freedom of the observations is not enough to infer all parameters of the reduction function. To address this problem, we may leave the task to the prior terms in Equation (2) where they must include extra assumptions such as Occam's razor [22] to guarantee that parameters can always be determined. Alternatively, we can apply safe solutions to the problems that solving is unfeasible. This concept is derived from human navigation behavior where we tend to preserve the current moving direction when there are no available landmarks [23].

### D. Summary

Algorithm 1 presents the summary of the Conformative Filter. It continuously queries observations, learns the reduction function, locates and executes the best found solution. The inputs are an environment space $W$, size $T$ of the navigation sequence or a destination $o$ in the reduced space, a reduced space $M$, and the start position $m_0$ in the reduced space. The algorithm maintains the most likely state variables $v, u, m$ from the latest observed $s, w$, while $v', u', m'$ are used to store the previous state variables. Because we wish to keep only the most likely variables, we substitute Equation (1),(2) into Equation (4),(5),(6) to merge the prior terms in propagation into the update rules. We abstain from considering all possibility in $\int P(r|v,u,m,w)\,\mathrm{d}r$ by introducing the variable $r$ that is used to store the expected to be seen features from $m$ or $v, u, w$.

## IV. Navigation using an inaccurate map

We propose an implementation of a Conformative Filter where the system is provided with an inaccurate map $M$ along with the path as the solution $Q$ to guide an agent.

**Algorithm 1** Conformative Filter

**Require:** $W, M, T, m_0$
  $v', u', m' \leftarrow$ null
  $m \leftarrow m_0$
  **for** $t = 0$ **to** $T$ **do**
    $r \leftarrow$ get features from $M$ at $m$
    $s, w \leftarrow$ query observations from $W$
    Ensure observability $s, w$
    $v \leftarrow \arg\max_v P(v|m, v', m')P(s|v, w)$
    **repeat**
      $u \leftarrow \arg\max_u P(u|m, u', m')P(r|u, v, m, w)$
      $r \leftarrow$ transform $w$ using $u, v$
      $m \leftarrow \arg\max_m P(r|m, v, u, w)$
      $r \leftarrow$ get features from $M$ at $m$
      $v \leftarrow \arg\max_v P(v|m, v', m')P(s|v, w)P(r|v, u, m, w)$
    **until** $v, u, m$ converge
    Execute solution from $M$ at $m$
    $v', u', m' \leftarrow v, u, m$
    $m \leftarrow \arg\max_m P(m|m')$
  **end for**

Because correct maps are too strict and harder to obtain, it is more flexible to create a system that can interpret inaccurate maps. We assume that the given map is a reduced space that has $n \in \mathbb{N}$ dimensions equal to the world $W$. The explicit part $v$ and the implicit part $u$ of the reduction function correspond to the set of recognized features and the location mapping between the world and the map respectively. The recognized features are the features that their types within the set $F = R_v$ in the map can be detected in the world, and the explicit part of the reduction function is the feature recognition process. After we have the type correspondences between the world and the map, we can use this information to guess the implicit part of the function by finding the best match features between the two domains (to put pins into possible locations) and solve it using regression. We apply local regression to learn the implicit part where all sub-locations have a linear property. This concept is analogous to non-rigid registration using a triangle mesh [24]. Especially in [14], they present this idea for multi-robot plan adaptation, which mainly focuses on matching the environment to discrete plans. We modify ideas presented in these papers for learning the reduction function and integrate state transition model to complete a Conformative Filter. Figure 3 sketches the idea of non-rigid registration for navigation using an inaccurate map.

### A. Learning the implicit part

Feature locations are assumed to be points in the world. To implement the local regression, we place a rectangle mesh of $b \in \mathbb{N}$ mesh points on the sensor space and relocate the features into this mesh's domain using barycentric coordinate to the three closest mesh points as shown in Fig 3. These coordinates are always fixed to the mesh. We can match

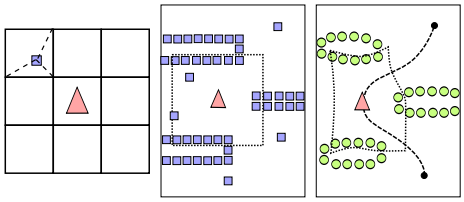


Fig. 3. Non-rigid registration for navigation using an inaccurate map. The left frame shows a rectangle mesh placed on the agent's (red triangle) sensor space. We use such a mesh as the implicit part of the reduction function; the middle and the right frame present its (functional) domain in the world and its codomain in the map respectively.

locations of the features to the map correspondences by positioning the mesh points in the map that optimizes the following equation [14]:

$$\hat{X} = \arg\min_{X} \underbrace{(L - BX)^T C (L - BX)}_{\text{location}} + \lambda \underbrace{X^T K^T K X}_{\text{prior}} \quad (7)$$

This equation serves as the learning rule for the implicit function. It has the location term and the prior term, which is equivalent to Equation (5). Suppose that there are $a \in \mathbb{N}$ features, $X$ is a $bn \times 1$ matrix contains map locations of the mesh points. $L$ is an $a \times 1$ matrix contains map locations of the features. $B$ is an $a \times bn$ matrix that keeps barycentric coordinates of the features. The $a \times a$ normalized weight matrix $C$ contains the degrees of feature match between feature pairs. It can also present the covariance of the feature locations along major axes. $K$ is a $bn \times bn$ distortion matrix defined by discrete Laplacian, which tend to produce low value when locations of the mesh points are symmetric [24]. It acts as the prior distribution in Equation (2) to guarantee observability of the system. Nevertheless, a minimum number of features is still required to solve the equation because $K^T K$ is rank deficient from the absence of position, rotation, scaling, and shearing. $\lambda \in \mathbb{R}^+$ is a weighting constant that defines degree of contribution between the two terms. Because the equation is in a quadratic form, the optimal mesh point locations are given by the following equation:

$$\hat{X} = (B^T C B + \lambda K^T K)^{-1} B^T C L \quad (8)$$

### B. Refining the explicit part

The feature recognition process is not always certain. It should be refined every time we have a new location cue. We define the recognition distance between the $i$-th feature in the world $f_{wi}$ and the $j$-th feature in the map $f_{rj}$ as below:

$$d_{ij} = \underbrace{(1 - P(f_{wi}, f_{rj}))}_{\text{distinctiveness}} + \underbrace{\frac{||g(l_{wi}) - l_{rj}||^2}{\sigma}}_{\text{location}} \quad (9)$$

This equation is equivalent to Equation (4) without the prior term. $P(f_{wi}, f_{rj})$ measures the similarity between feature $f_{wi}$ and $f_{rj}$. The first term is minimized if and only if the maximum of $P(s, r|v, u, m, w)$ is reached when considering the types of features. The second term is equivalent to $P(r|v, u, m, w)$ when considering only locations. $l_{wi}, l_{rj} \in \mathbb{R}^n$ are the locations of the features. $g : \mathbb{R}^n \to \mathbb{R}^n$ is location transform function from the world to the map using the learned mesh. $\sigma \in \mathbb{R}^+$ is used to control weight between the distinctiveness and the location term. We want to choose matches from the world and the map that the minimize sum of such functions between those pairs. If matching is one-to-one, this is an assignment problem and the solution is given by the Kuhn-Munkres algorithm [14], otherwise each pair can be chosen independently. We also provide a maximum distance $d_{max} \in \mathbb{R}^+$ as the criteria to classify missing features. A feature match is discarded when its best computed distance is greater than $d_{max}$.

The transition between states can also help learning the reduction function. A known propagation model, from odometry for example, can be incorporated into the recognition process to find the initial matches as suggested in Equation (6). At any state, we can track features from the previous state using the following distance function:

$$\tilde{d}_{ij} = (1 - P(f_{wi}, f_{w'j})) + \frac{||l_{wi} - h(l_{w'j})||^2}{\psi} \quad (10)$$

Here we try to minimize the sum of distances from the previous state to the current state where $f_{w'j}$ is the $j$-th feature in the previous state. $l_{w'j}$ is its location in the world. $P(f_{wi}, f_{w'j})$ measures how close the current world feature is to $f_{w'j}$. $h : \mathbb{R}^n \to \mathbb{R}^n$ is the state transition function. and $\psi \in \mathbb{R}^+$ is a constant similar to $\sigma$. We can use the same optimization routine for Equation (9). The best previous matched features are then assigned to their correspondences in the current state.

### C. Summary of navigation using an inaccurate map

The process of navigation using an inaccurate map with a Conformative Filter follows Algorithm 1. For a navigation task in an environment $W \subset \mathbb{R}^n \times F$ with a map $M \subset \mathbb{R}^n \times F$, the system runs with a starting position $m_0 \in \mathbb{R}^n$ until the agent reaches $m_T \in \mathbb{R}^n$. At each step, the system predicts the current state from the previous control. Then, it queries the set of world features $F_w \subset F$ from sensor reading and fills the set of map features $F_m \subset F$ by choosing features that expected to be seen from the current state. $F_w, F_m$ and Equation (9),(8) are used to refine the feature matching and learn the mesh point locations. These equations implement the E and the M step from the previous section, therefore, they must be alternatively iterated. In this case, the number of iterations is fixed to $K \in \mathbb{N}$ to guarantee termination. Equation (10) is used to find the initial matches before the iteration starts. During the iterations, $\sigma$ and $\lambda$ can be fixed or gradually changed similar to the concept of the temperature scheduling in simulated annealing [24]. Once the mesh point locations are available, we can use them to compute the agent location in the map and obtain the solution path. Finally, the agent follows the path and the process continues to the next step.

## V. EXPERIMENTS

We evaluate the algorithm in navigation tasks using a hand-drawn map where the distant metric is distorted and irrelevant details are missing. The experiments are set on various simulated 2D worlds. The features are corners, lines, and endings. Each feature contains position, orientation, and surface normal. The latter is used to determine the chance of detection in the map for filling in $F_m$. The similarity between features is defined by the dot products of the orientation vectors, though we subtract 50% penalty from all pairs of different type features. These features can be seen in both the world and the map as the observations. For the map, the system operator can effortlessly place them via our program interface created specifically for the tasks. In the world, the navigation agent is simulated in Player [25] with an omnidirectional laser scanner. The agent itself is non-holonomic with degrees of freedom in position and orientation. It can also provide the prediction of movements in the world that can be used to propagate the possibilities of these configurations. The spaces of configuration, measurements, and the feature location in the world are homogeneous, thus they can be transformed directly into the map space using the same learned function, which is presented as a rectangle mesh in our case. This information is used to compare with the preferred path, which is drawn to the map using the same program interface, to obtain the solutions to control the agent in forms of movement commands: moving forward or turning.

We compare the results from our algorithm and a rigid localization that assumes no location distortion between the worlds and the maps. The rigid localization acts as a representation for conventional navigation approaches without learning the location distortion function, but it also contains the refinement process for feature recognition with the same number of iterations. The result is two-fold. First we show the performance qualitatively from a set of selected examples, then we define metrics to measure the performance quantitatively.

### A. Qualitative comparison

Figure 4 shows the result paths that generated from both algorithms in five environments. All of the maps are drawn so that the start regions are identical to the worlds to serve as the control part. Because the rigid localization requires the identical maps to work perfectly, we show that the maps are not biased, our algorithm can navigate in the accurate maps as well, and how both algorithms perform differently at the middle of the runs. The numbers of iterations for learning the reduction function in both algorithms are fixed to 2 and we set the schedule for $\sigma$ to $4, 1$ time the width of the narrowest free area. The maximum distance for classifying missing features is set to $\frac{\sqrt{2}}{2} + 1$, which means a feature is discarded when it and its best match differ more than $\frac{\pi}{4}$ degree of direction vectors and are further away than $\sqrt{\sigma}$ from each other. The $\psi$ is assigned to 1 assuming that the agent moves slowly relative to the process time used in each step. We assign a $\lambda$ for each map empirically by qualitatively comparing the paths generated from the set $\{1, 0.5, 0.1, 0.05\}$ and choose the best which is shown in Figure 4. The size of the rectangle mesh is 441 points covering the entire laser scan range.

In the first world, the path from the rigid localization deviates much further from the path from our algorithm. Such deviations usually cause the agent to collide into the obstacles in the third world. For the second world where we want the agent to perform area coverage, the result shows that our algorithm can fit the map to the world better than the rigid localization and thus produces a better coverage path. In the fourth world, the agent from the rigid localization never reaches the destination. The paths in the fifth world are almost identical, but there are position-shifting in the path from the rigid localization that can cause navigation failures when the agent moves through the scaled hallway.

### B. Quantitative comparison

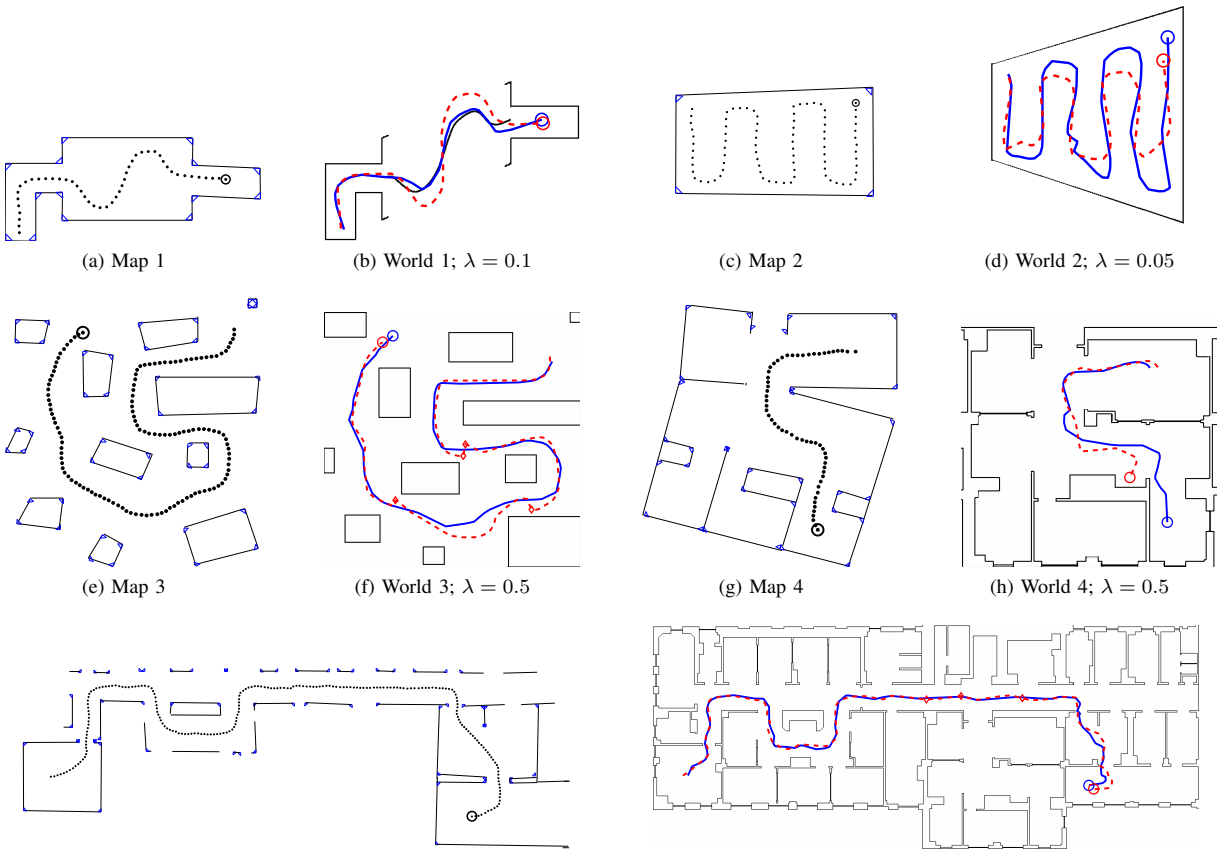
In this part, we first show the performance of the Conformative Filter in a form of closeness between paths. The closeness is measured from the average closest Euclidean distances from all points in the generated paths to the preferred one. We use the first world in Figure 4 where the amount of distortion is known. The preferred paths in the world are projected from the maps using the known shearing amount. Second, we create a simple coverage task in the second world and measure the average closest Euclidean distances from all points in the covering area to the generated paths. We execute the algorithms 2 rounds for each of five slightly different maps and the result shows that the path distance from our algorithm with the best $\lambda$ in the first world is only 32.3% and the average coverage distance of our algorithm in the second world is 59.9% of the rigid localization's while the time used in each step is only 1.1% increased.

TABLE I
AVERAGE OPERATOR INTERVENTION COUNT FOR EACH MAP

| Approach/Map | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Conformative Filter | 0.6 | 0.3 | 1.5 | 0.5 | 0.6 |
| Rigid localization | 0.1 | 0.0 | 2.4 | 1.3 | 1.1 |

Finally, we measure the average number of times that we override the process in order to keep the agent on the preferred path. [2] The system is intervened when the agent's locations from the world and the map differ or when the agent is too close to an obstacle based on the human operator's decision. To make the experiment unbiased for each run, we use the same operator and choose an algorithm from the two while hiding its identity from the operator. In real scenarios, the $\lambda$ in our algorithm is not provided and need to be tuned, thus we include all of the interventions when tuning $\lambda$ into consideration. We run our algorithm 2 rounds per each $\lambda$ and 2 more rounds for the best $\lambda$ on each world. The result is shown in Table I. In the first and the second world, the number from our algorithm is higher because the mesh is folding when

[2] In reality, the system is fully autonomous during the execution. We introduce the concept of operator overriding for the evaluation purpose only.



(a) Map 1 (b) World 1; $\lambda = 0.1$ (c) Map 2 (d) World 2; $\lambda = 0.05$

(e) Map 3 (f) World 3; $\lambda = 0.5$ (g) Map 4 (h) World 4; $\lambda = 0.5$

(i) Map 5 (j) World 5; $\lambda = 0.5$

Fig. 4. Experiment results. The preferred paths are shown as the black dotted lines in the maps. The paths taken by the agent according to the Conformative Filter and the rigid localization are shown in the blue lines and the red lines respectively. The circles indicate the destinations, and the red diamonds show where the operator interventions occur in rigid localization. The black solid line in the first world is the projected preferred path.

it converges to a wrong place and the agent usually collides into the obstacles given specific $\lambda$s. The rigid localization often drives the agent out of the path in the third and the fourth world where the smaller number from our algorithm comes from the folding mesh. In the fifth world, most of the fixings come from the position-shifting when the agent uses the rigid localization. Diamonds in Figure 4 shows where the interventions occur.

### C. Analysis

In the experiments, we observe both advantage and disadvantage of the Conformative Filter. For the rigid localization, though it performs generally well in navigation, it sometimes deviates the agent from the preferred path and can cause position-shifting. Because the distortions are not identical for all locations in the map, the rigid localization tries to minimize the error from all observed locations with respect to its strong rigid assumption, yielding poor localization results, while the Conformative Filter allows more flexible one, and thus resulting in better path quality, which can be applied in certain applications such as area coverage. Furthermore, the process time for our algorithm is comparable to the rigid localization's because the function is learned in a parametric form. This shows that the task of conformation does not adding too much load to the system while it yields a better result. The disadvantage of our algorithm is that there is no guarantee optimal solution because it is based on Expectation-maximization algorithm. Also, the learned function sometimes converges to a wrong place depending on the choice of learning parameters and the starting mesh shape. This phenomenon is observed when the learned mesh is folding and it causes the agent location diverged. It can be prevented by employing a stronger prior assumption for the mesh, or simply resetting the mesh shape when the folding occurs. Nevertheless, both algorithms sometimes cause the agent to collide into the obstacles. In real scenarios, hierarchical control can be used to avoid this failure; for example using obstacle avoidance instead of the current control when the sensor reading indicates that the agent is too close to an obstacle.

## VI. Discussion

For a navigation problem using an inaccurate map, the Conformative Filter provides the solver system with a reduction

function to evaluate the likeliness between the environment and the map. Learning the function also has another benefit, that is when we want to transform some quantities between the two domains. For example, in a navigation task we want to find the agent's location in the map, and sometimes solutions from the map, such as places to go, need to be transformed into the world before the agent can adopt them.

Instead of finding only one best location as suggested in Algorithm 1, condensation-based methods can be used to maintain the set of possible locations. However, searching can be costly because the process time is known to increase exponentially to the dimension of the reduction functions. We can alternatively maintain the variance of locations which can be done efficiently when both propagation and update rules are linear.

We evaluate our algorithm on simulated worlds to reduce the difficulties of the experiments when dealing with real agents. On the other hands, the simulations verifies the concept for the real worlds because we try to localize the agent in reduced spaces that have the same difficulties in the real scenarios and the simulations under the absence of the reduction functions. In particular, we do not assume any characteristics of the real worlds or the simulated worlds in the main algorithm except that they should satisfy the Markov property which has been proven successful in both scenarios [18],[21].

In terms of contributions, we present it in two aspects. From the Bayes' filter point of view, we extend the basic approach with the idea of conformation to create the Conformative Filter and compare their results in navigation tasks using a hand-drawn map. The results show that our approach outperforms the conventional filter without conformation. From the conformation aspect, we use Bayes' filter to find initial setting for the Expectation-maximization part of the process. It also helps truncate the necessity to search over the entire solution space, which cannot be done efficiently if the space is continuous. Unlike in [14], their solution space is discrete and thus smaller than our examples in the experiments.

## VII. Conclusion

We present the Conformative Filter for solving navigation problems that satisfy the Markov property using inaccurate maps. The algorithm is based on the idea of reduction into a reduced space, which is used to present an inaccurate map with a solution path. In every step, the algorithm attempts to locate the current state in the map while trying to infer both the explicit and the implicit part of the reduction function in Expectation-maximization manner. Bayes' filter is used to limit the search scope for localization and initialize the starting value for EM. We also provided an example implementation for the Conformative Filter using local regression. The evaluation is done on the simulated worlds with a set of hand-drawn maps. It clearly shows the benefits of the Conformative Filter which improves localization in the maps and maintains the same amount of load when compared with Bayes' filter using a rigid localization.

## References


[1] M. Tomono and S. Yuta. Indoor navigation based on an inaccurate map using object recognition. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 619–624. IEEE, 2002.

[2] V. Setalaphruk, A. Ueno, I. Kume, Y. Kono, and M. Kidode. Robot navigation in corridor environments using a sketch floor map. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on*, volume 2, pages 552–557. IEEE, 2003.

[3] G. Chronis and M. Skubic. Robot navigation using qualitative landmark states from sketched route maps. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1530–1535. IEEE, 2005.

[4] M.M. Veloso. *Planning and learning by analogical reasoning*. Springer, 1994.

[5] B. Nebel and J. Koehler. Plan reuse versus plan generation: a theoretical and empirical analysis* 1. *Artificial Intelligence*, 76(1-2):427–454, 1995.

[6] M. Fox, A. Gerevini, D. Long, and I. Serina. Plan stability: Replanning versus plan repair. In *Proc. ICAPS*, 2006.

[7] L. Spalzzi. A survey on case-based planning. *Artificial Intelligence Review*, 16(1):3–36, 2001.

[8] A. Blake and A. Zisserman. Visual reconstruction. 1987.

[9] S. Simhon and G. Dudek. Analogical path planning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 537–543. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[10] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in cognitive sciences*, 3(6):233–242, 1999.

[11] Y. Wu and Y. Demiris. Towards One Shot Learning by imitation for humanoid robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2889–2894. IEEE, 2010.

[12] A. Billard and M.J. Matari. Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 37(2-3):145–160, 2001.

[13] Aaron P. Shon, Keith Grochow, Aaron Hertzmann, and Rajesh P. N. Rao. Learning shared latent structure for image synthesis and robotic imitation. In *In Proc. NIPS*, pages 1233–1240. MIT Press, 2006.

[14] B. Takács and Y. Demiris. Multi-robot plan adaptation by constrained minimal distortion feature mapping. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 742–749. IEEE, 2009.

[15] W. S. Havens and Alan K. Mackworth. Schemata-based understanding of hand drawn sketch maps. In *Proc. 3rd CSCSI Conf.*, Victoria, BC, 1980.

[16] W.K. Yeap. Towards a computational theory of cognitive maps. *Artificial Intelligence*, 34(3):297–360, 1988.

[17] C.M. Bishop and SpringerLink (Online service). *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[18] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11(3):391–427, 1999.

[19] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

[20] S. Granger and X. Pennec. Multi-scale EM-ICP: A fast and robust approach for surface registration. *Computer VisionECCV 2002*, pages 69–73, 2006.

[21] J. Diard, P. Bessiere, and E. Mazer. A Survey of Probabilistic Models Using the Bayesian Programming Methodology as a Unifying Framework. 2003.

[22] D. Donoho, H. Kakavand, and J. Mammen. The simplest solution to an underdetermined system of linear equations. In *Information Theory, 2006 IEEE International Symposium on*, pages 1924–1928. IEEE, 2006.

[23] P.E. Michon and M. Denis. When and why are visual landmarks used in giving directions? *Spatial information theory*, pages 292–305, 2001.

[24] J. Pilet, V. Lepetit, and P. Fua. Fast non-rigid surface detection, registration and realistic augmentation. *International Journal of Computer Vision*, 76(2):109–122, 2008.

[25] RT Vaughan, BP Gerkey, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference Advanced Robotics (ICAR 2003)*, pages 317–323, 2003.