Proceedings of the 2001 IEEE
International Conference on Robotics & Automation
Seoul, Korea · May 21-26, 2001

Vision Based Navigation for an Unmanned Aerial Vehicle

Bruno Sinopoli* Mario Micheli* Gianluca Donato† T. John Koo*

*Robotics and Intelligent Machines Laboratory, UC Berkeley, Berkeley, CA, USA
†DigitalPersona Inc., Redwood City, CA, USA

{sinopoli,micheli,koo}@eecs.berkeley.edu
gianlucad@digitalpersona.com

Abstract

We are developing a system for autonomous navigation of unmanned aerial vehicles (UAVs) based on computer vision. A UAV is equipped with an on-board cameras and each UAV is provided with noisy estimates of its own state, coming from GPS/INS. The mission of the UAV is low altitude navigation from an initial position to a final position in a partially known 3-D environment while avoiding obstacles and minimizing path length. We use a hierarchical approach to path planning. We distinguish between a global offline computation, based on a coarse known model of the environment and a local online computation, based on the information coming from the vision system. A UAV builds and updates a virtual 3-D model of the surrounding environment by processing image sequences and fusing them with sensor data. Based on such a model the UAV will plan a path from its current position to the terminal point. It will then follow such path, getting more data from the on-board cameras, and refining map and local path in real time.

1 Introduction

Without complete knowledge of the environment an agent can only plan a path which is optimal with respect to its knowledge at the time of planning. Based on multiresolution environmental models we use a hierarchical approach to path planning, with different paths designed at different time and space scales. Our approach takes its move from the work of Reissell and Pai [5], who propose a path planning scheme based on multiresolution terrain representation. In our approach we divide the path planning in two parts: a global offline computation, based on a coarse model of the environment and a local online computation, based both on the original model and on the information provided by the vision system.

Our testbed is an Unmanned Aerial Vehicle (UAV). The UAV makes use of an *a priori*, inaccurate, graph model of the terrain to plan an initial, coarse path. We use wavelets to filter the map to the desired level of abstraction. A few waypoints are selected based on the desired objective. At this level of abstraction we perform a global offline computation on the entire graph. Computation of optimal path over the complete terrain model is very intensive. At this stage, the planning is performed deterministically. We use standard optimization algorithms for shortest path computation, such as *Djikstra* or A^*.

On the other hand, in-flight navigation mainly depends on the information gathered by the vision system. We propose a probabilistic approach to local online path planning, for a number of reasons: first of all, because of the inevitable uncertainty of measurements from the sensors; secondly, for the intrinsic uncertainty of an unknown surrounding environment; and finally, the structure of reasoning of any (biological or artificial) intelligent system is naturally probabilistic—whenever a decision has to be taken, the costs or gains that all possible choices imply are "weighed" in probabilistic terms, and the decision that is more "likely" to yield maximum gain is taken.

The surrounding three-dimensional environment is divided into cells. Initially each of the cells is assigned with a probability of occupancy. We will call such probability function a "risk map"—a risk map value close to one indicates high risk (presence of an obstacle), while a value close to zero denotes low risk (no obstacle). Such an approach was introduced by Thrun [15]: in this work we extend it to three dimensional environments using vision rather than sonar sensors. The UAV is equipped with an initial knowledge of the surrounding environment through an a priori risk map assigned from the mission planner. However, such a risk map will be refined by the UAV during navigation exploiting sensor data (i.e., multiple image sequences and state data containing UAV's position, orientation, velocity, etc.).

Processing multiple image sequences and integrating such information with other sensor readings allows a UAV to estimate the distance between itself and the obstacles, and provide a measure of the uncertainty of the estimates (in terms of error variances). Using all past information in an "optimal" way, the UAV is able to refine its virtual map of the environment, and thus obtain a model that is more accurate and up-to-date.

Given a probabilistic model of the environment, path planning can be performed using dynamic programming techniques to plan a discrete path, as a sequence of adjacent cells. Each cell corresponds to a state of a stochastic transition system, and a cost is assigned to each state transition. The final objective is to minimize the total expected cost.

The next section will describe the system architecture and model. Sections 3 and 4 will provide a detailed description of our navigation algorithm for both offline and online computation. The last section is devoted to conclusion and comments.

2 System Architecture

Our testbed is a helicopter-based UAV, part of a research project undertaken at UC Berkeley under the acronym of BEAR (Berkeley Aerobots) [1]. Compositional methods represent a natural way to reduce complexity of system design, by decomposing the problem into a sequence of smaller problems of manageable complexity. Hierarchy allows to separate complex global task in a series of simpler, local ones. The helicopter is modeled as a hierarchical hybrid system. For a detailed discussion please refer to [14]. The system is inherently hybrid, having to combine continuous control with discrete logic. The helicopter model consists of three components: the Flight Management System (FMS) which is responsible for planning and controlling the operation of the UAV, the vision system for the detection and investigation of objects of interest and the helicopter, i.e. the vehicle dynamics. The FMS consists of four layers, the strategic, tactical, and trajectory planners, and the regulation layer, as described in Figure 1.

The **Strategic Planner** is concerned with the planning and execution of the central UAV mission. It designs a coarse, self-optimal trajectory, which is stored in form of a sequence of waypoints. This layer also takes care of the transition between the points, by acknowledging the completion of a subtask and scheduling the next one.

The **Tactical Planner** is responsible for local obstacle avoidance: it plans a discrete trajectory between the waypoints provided by the Strategic Planner and must modify it on-line in real time in case of appearance of new obstacles along the

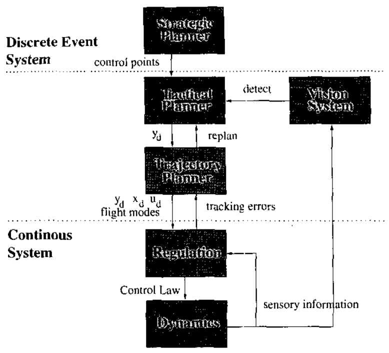

Figure 1: System Architecture

previously planned path. To do this, it makes use of data provided by the camera, GPS and internal sensors on position, orientation, linear and angular velocities.

The **Trajectory Planner** interpolates the set of points into a continuous trajectory, that the lower layers of the system will have to follow. Such trajectory will have to be *trackable*, i.e. compatible with the UAV's dynamics. In safety critical situations the Trajectory Planner might overrule the behavior proposed by the Tactical Planner, and send to the system's lower layers continuous trajectories that correspond to safety manœuvres.

The **Regulation Layer** and the **Dynamics Layer** represent the continuous control part of the system. Their study is out of the scope of this paper, as is the one of the Trajectory Planner. We suggest reading papers [7] and [6] for a detailed description of the system's lower layers and various control designs.

In the next section we shall describe our approach to path planning. We design both a global offline and a local online navigation scheme.

3 Global Navigation: the Strategic Planner

In 3D navigation the choice of an appropriate model for the surrounding environment is crucial. In our design we make extensive use of Digital Elevation Models (**DEM**) [2]. Recent advances in laser technology have provided us with an extensive coverage of earth surface with extreme level of accuracy. The basic idea consists in gridding the surface and assigning an altitude to each cell in the grid. The gridding is up to $1m$ with accuracy in the range of centimeters. These models are widely used in Earth Sciences. In our approach we manipulate these models and use them at different levels of resolution via extensive use of wavelet transform.

3.1 The Wavelet transform

Following the approach used in [5] we choose to perform a wavelet decomposition with the *pseudocoiflet* family introduced by [13]. The wavelets properties (and those of the pseudocoiflets family in particular) that justify their use as opposed to other multiresolution filtering techniques are their *Space–frequency localization*, which provides local information about the smoothness of the data, and their *natural hierarchical representation.* Wavelets also provide the *best possible approximation for continuous functions in* L^2. The approximation error can be estimated precisely; the theorem of vanishing moments provides a way to relate the error to the properties of the mother wavelet. In the pseudocoiflet case the number of vanishing moments is 4, therefore the error decreases as 2^{4l} as scale l increases (i.e. we examine the data at finer scales).

3.2 Terrain analysis for flight planning

Using the classic wavelet notation we can write the wavelet decomposition of a one dimensional signal as a repeated application of two finite filters H and G:

$$\begin{tikzcd} s \arrow[r, "H", "G"'] \arrow[d] & s_1 \arrow[r, "H", "G"'] \arrow[d] & s_2 \arrow[r, "H", "G"'] \arrow[d] & \dots \\ w_1 & w_2 & \dots & \end{tikzcd}$$

where the sequences s_l and w_l are respectively the scaling and wavelet coefficients at scale l. Given the complete set of wavelet coefficients w_{lj} –where j indexes the position– reconstruction is performed using the corresponding biorthogonal filters $\tilde{H}$ and $\tilde{G}$.

For a two-dimensional signal the decomposition generates three sequences that correspond to the horizontal, vertical and diagonal details of the image, as shown in figure 2.

Due to the pseudocoiflets properties the scaling coefficients form a sampling of approximation surface that is smoother than the original approximation surface, as shown in Figure 3.

3.3 Algorithm and Results

We employ *Dijkstra's* algorithm to find the shortest path on the transformed grid. Our cost function result from the sum of three factors, appropriately scaled. Every cell point has a cost associated with it:

$$C(i,j) = \alpha_1 c_d(i,j) + \alpha_2 c_e(i,j) + \alpha_3 c_h(i,j), \quad (1)$$

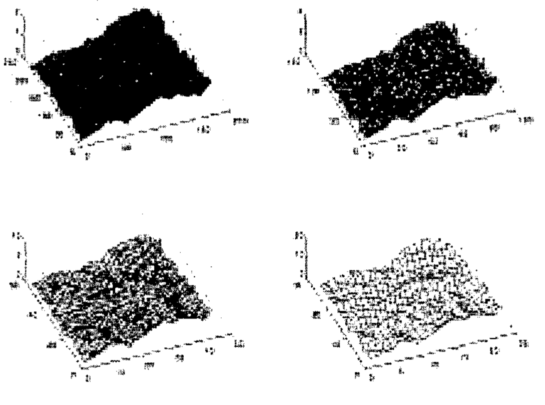
Figure 3: The Terrain at different levels of detail

where c_d, c_e, c_h are costs associated with distance to goal, roughness of terrain [5], and flight altitude respectively. The coefficients α_i will assign a particular weight to each cost. The UAV will look for the shortest path on a smooth part of the terrain with low altitude. Figure 4 shows a typical outcome of this algorithm. In the figure the waypoints have been interpolated.

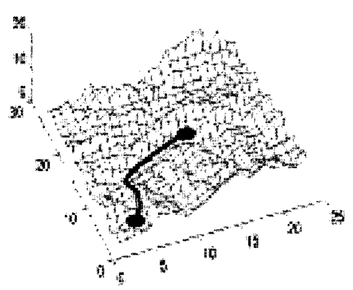
Figure 4: Path on a DEM of La Honda, CA.

4 Local Navigation: the Tactical Planner

In this section we shall focus on the problem of vision-based local obstacle avoidance, which is the task of the Tactical Planner. We shall describe the strategy formulated by M. Micheli in [11].

Given a set of waypoints, provided by the Strategic Planner, the Tactical Planner connects them with a discrete, finer trajectory, i.e. a set of control

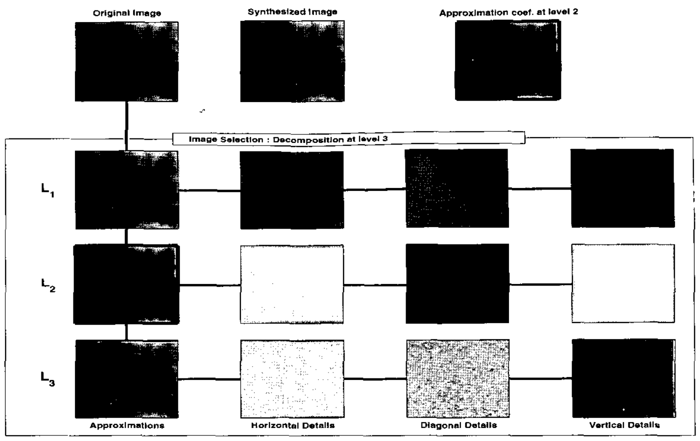

Figure 2: Decomposition of altitude profile of La Honda, CA

points in three dimensional space which will successively be interpolated into a continuous, trackable trajectory by the Trajectory Planner (see Figure 1). We shall assume that the UAV is provided with an on-board computer, one or more cameras, a GPS system and sensors that give instantaneous, noisy estimates of the agent's orientation in space and (three-dimensional) linear and angular velocities.

4.1 Bayesian Three-Dimensional Occupancy Grid Building

The main idea the whole strategy is based on is splitting local space (i.e. a connected "neighborhood" of the two specific waypoints that we are considering) into three-dimensional *cubic cells* (see Figure 5). To each cell we associate a *grid point*, which is simply the center of a cell; grid $\mathcal{G}$ is the set of all grid points. We assume that the agent's initial position coincides with a specific grid point, which corresponds to the first of the two way points that were previously provided by the Strategic Planner. We also assume that the second waypoint coincides with another grid point. The Tactical Planner's task is to provide a succession of adjacent grid points (or cells) that connect the agent's initial position and the second waypoint, i.e. a *discrete* trajectory connecting the two waypoints. Such trajectory will have to avoid locally detected obstacles, and must be modified on-line and in real time in case of abrupt environmental changes, e.g. the sudden appearance of new obstacles. Furthermore, the Tactical Planner should account for the constraints on the Trajectory Planner that will have to interpolate the discrete trajectory into a continuous, trackable one, which will have, among others, curvature radius constraints.

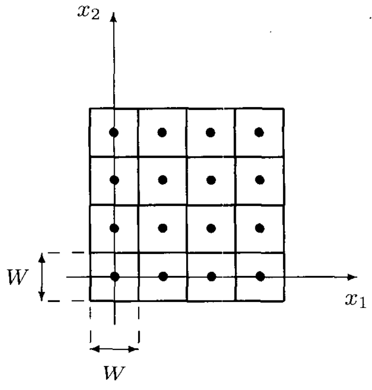

Figure 5: Two-dimensional example of grid: dots represent grid points and squares represent cells.

The Risk Map. Each cell may belong to one of two classes: *occupied* or *not occupied* by an obstacle; we shall indicate this class set as $\mathcal{C} = \{\text{occ}, \overline{\text{occ}}\}$. Given a probability space $\mathcal{S} = (\Omega, \mathcal{F}, P)$, for each grid point $Q \in \mathcal{G}$ we will consider $C(Q)$ as a *random variable*, i.e. a function $C(Q) : \Omega \to \mathcal{C}$. Therefore, for all $Q \in \mathcal{G}$, $C(Q)$ has a pre-assigned probability of belonging to class *occupied*: $P[C(Q) \in \text{occ}]$; we'll call this function of Q the *a priori probability of occupancy*. Its value will be suggested by the prior knowledge one has on the local environment: e.g. a value of 0.5 indicates no knowledge, or maximum entropy (in the information-theoretic sense).

The agent continually receives new data about the local environment through its sensors, e.g. the on-board digital camera. We shall indicate with $\mathbf{D}_k(Q)$ the data vector relative to the k-th measurement at point $Q \in \mathcal{G}$. $\mathbf{D}_k(Q) : \Omega \to \mathbb{R}^N$ is a *random vector*; being a function of index $k \in \mathbb{Z}^+$ it can be viewed as an N-dimensional *stochastic process*.

Define the *k-th risk map* as the following function of data:

$$R_k(Q; \mathbf{d}_k, \ldots, \mathbf{d}_1) := \\ P\big[C(Q) = \text{occ} \bigm| \mathbf{D}_k(Q) = \mathbf{d}_k, \ldots, \mathbf{D}_1(Q) = \mathbf{d}_1\big] ;$$

the risk map associates to each grid point $Q \in \mathcal{G}$ the probability that the corresponding cell is occupied by some obstacle, given all the first k measurements relative to that cell; a probability close to one indicates high "risk", whereas a value close to zero suggests the cell, according to the information we have, is "probably" free. Introducing the compact notation: $\mathbf{D}^k(Q) := [\mathbf{D}_k(Q), \ldots, \mathbf{D}_1(Q)]$, $\mathbf{d}^k := [\mathbf{d}_k, \ldots, \mathbf{d}_1]$, the risk map definition may be rewritten as follows:

$$R_k(Q; \mathbf{d}^k) := P\left[C(Q) = \text{occ} \bigm| \mathbf{D}^k(Q) = \mathbf{d}^k\right] ;$$

in particular, the risk map calculated at $k = 0$ coincides with the *a priori* probability of occupancy: $R_0(Q) = P\left[C(Q) = \text{occ}\right]$.

We will now study the evolution of the risk map (for growing values of time index k) in function of the following conditional probabilities:

$$P\big[C(Q) = \text{occ} \bigm| \mathbf{D}_i(Q) = \mathbf{d}_i\big] , \qquad 1 \le i \le k; \quad (2)$$

which we shall call *risk map updating functions* (for reasons that will be clarified); they represent the probabilities of occupancy of the cell corresponding to Q given the value that the *single* measurement vector $\mathbf{D}_i(Q)$ takes. We shall assume that measurements about any single cell are *conditionally independent* given the state of the cell, i.e.:

$$P\left[\mathbf{D}_1(Q) \in B_1, \ldots, \mathbf{D}_n(Q) \in B_n \,|\, C(Q) = c\right] = \\ = \prod_{j=1}^{n} P\left[\mathbf{D}_j(Q) \in B_j \,|\, C(Q) = c\right] , \qquad (3)$$

where $\{B_1, \ldots, B_n\}$ are arbitrary Borel subsets of $\mathbb{R}^N$ and $c \in \{\text{occ}, \overline{\text{occ}}\}$ is the state of the cell corresponding to grid point Q.

We shall now focus our attention on a particular grid point, therefore we shall simply write $R_k(\mathbf{d}^k)$ instead of $R_k(Q; \mathbf{d}^k)$, $P(\text{occ})$ instead of $P[C(Q) = \text{occ}]$, and $P(\text{occ} \,|\, \mathbf{d}^k)$ instead of $P[C(Q) = \text{occ} \,|\, \mathbf{D}^k(Q) = \mathbf{d}^k]$. It is possible to prove [11] [15] that, given hypothesis (3), the following *risk map updating law* holds:

$$R_k(\mathbf{d}^k) = 1 - \left\{ 1 + \frac{P(\text{occ} \,|\, \mathbf{d}_k)}{1 - P(\text{occ} \,|\, \mathbf{d}_k)} \cdot \right. \qquad (4) \\ \left. \cdot \frac{R_{k-1}(\mathbf{d}^{k-1})}{1 - R_{k-1}(\mathbf{d}^{k-1})} \cdot \frac{1 - P(\text{occ})}{P(\text{occ})} \right\}^{-1} ,$$

which has to be initialized setting $R_0 = P(\text{occ})$. Clearly, function $P(\text{occ} \,|\, \cdot\,) : \mathbb{R}^N \to [0, 1]$, i.e. the risk map updating function (its name is now justified) plays a fundamental role. Such function is actually unknown; according to the *nature* of incoming data (in our case, an image sequence and position/velocity information), it has to be approximated in some way. In other words, we have to build an appropriate function of data that approximates the probability of cell occupancy after a single observation on the cell has been performed.

Other authors have used the grid-based method for environment modeling, but always in two dimensions, i.e. for indoor robot navigation [12] [15]. For example, Thrun [15] assumes the ground robot is equipped with an array of sonar sensors; he then constructs the grid map updating function through an artificial neural network, training it with examples of sensor readings. As we will illustrate later on, we have found an *analytic expression* for the risk map updating function, which also accounts for uncertainties (noise) in our sensor readings (cameras, GPS, etc.).

Remark. In implementing our strategy *cell size* is clearly a very important parameter. It should be chosen in accordance with the local environment's size, the type of the environment (i.e. the type and size of typical obstacles in the environment), the agent's size and the on-board computer's computational power and storage capabilities, keeping in mind that incoming data must be processed in real time.

It is up to the engineer's experience and knowledge to choose an appropriate cell size. For example, making the grid too fine with respect to the size of the agent or the typical obstacles that are present in the environment simply wouldn't make sense, unless we could count on a very powerful onboard computer; on the other hand, making the grid too coarse would be an inefficient use of incoming data.

4.2 Vision-based Risk Map Updating

Consider the geometric model illustrated in Figure 6. The UAV is equipped with one on-board

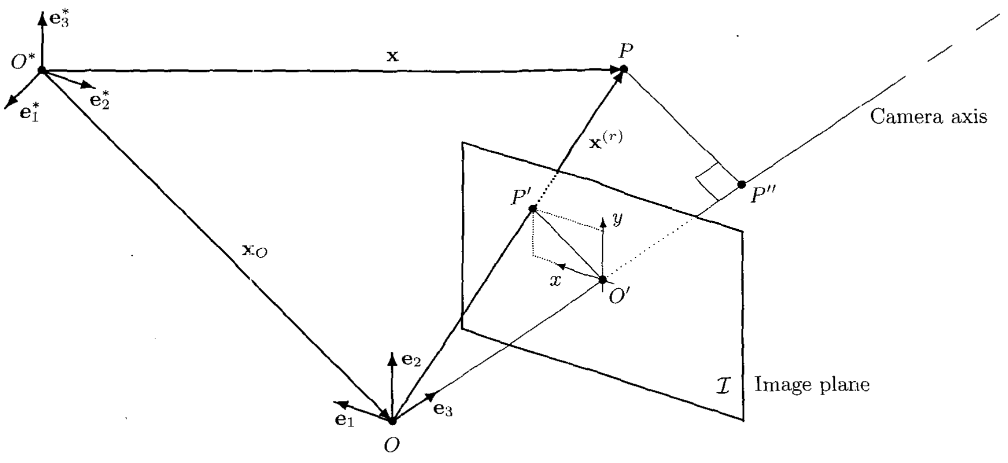

Figure 6: Camera model. Body frame $\mathcal{B} = \{O(t), \mathbf{e}_1(t), \mathbf{e}_2(t), \mathbf{e}_3(t)\}$ is attached to the camera in *rigid motion.*

camera, that provides it with an image sequence; we consider a reference frame (*body frame*) in *rigid motion* with the agent (and the camera). As we specified at the beginning of this section, we assume that the UAV is also provided with noisy estimates of its position with respect to an *inertial frame* (by GPS) and its three dimensional linear and angular velocities, with respect to the body frame.

We first developed an efficient multi-scale probabilistic algorithm [11] for *optical flow* recovery (see also [8], [9]); we then modified it to obtain a technique that allows the simultaneous recovery of optical flow and *depth* (see also [10]) in an optimal way, i.e. making the best use of all past data. The algorithm provides, for each pixel of the *depth map*, the corresponding error variance, i.e. a quantity that somehow measures the amplitude of depth estimation error (i.e. the reliability of our estimate). Such quantity is a function of all the incoming data error variances.

The depth map provides information about the distance between the agent and the obstacles that appear on the image plane. Suppose we want to establish whether or not a cell is occupied by an obstacle, and that P is the grid point corresponding to that cell (see Figure 6); knowing the grid point coordinates and being provided with the agent's position (with respect to the inertial frame) it is possible to calculate the distance between the grid point and the agent, and to establish the *projection* of point P onto the image plane (point P' in Figure 6). Now, comparing such distance with the depth reported on the *depth map* at P', one is able to establish whether point P is in front of, behind, or on the surface of an obstacle. Calling s the difference between the value reported on the depth image at P' and the distance between P and the agent, we have found [11] the following expression for the risk map updating function:[1]

$$\widehat{P}(\text{occ}|\mathbf{d}_k) = R_0\,\Phi\Big(\frac{s}{\sigma}\Big) + (1-R_0)\frac{\xi}{\sqrt{\sigma^2+\xi^2}}\cdot \quad (5)$$
$$\cdot\exp\left\{-\frac{1}{2}\frac{s}{\sigma^2}\left(s-\frac{1}{\sigma^2+\xi^2}\right)\right\}\cdot\Phi\bigg(\frac{\xi}{\sigma}\frac{s}{\sqrt{\sigma^2+\xi^2}}\bigg),$$

whose shape is reported in Figure 7; R_0 is the *a priori* risk map value for the grid point we are considering, ξ is a parameter that is proportional to the typical depth of obstacles in the environment we are navigating into (it has to be set off-line, before navigation starts), and σ^2 is the error variance associated with s (defined above), which is a function of all the incoming data error variances; $\Phi(\cdot)$ is the normalized, zero-mean Gaussian distribution.

Note that if $s < 0$ and $|s|$ is sufficiently large (i.e. the cell is *in front* of the obstacle represented in P' on the image plane) then we set the posterior probability of occupancy virtually equal to zero; if $s \simeq 0$ (the grid point lies on the obstacle's surface) then we set such probability close to one; for

[1]The "hat" ($\widehat{\ }$) we use here distinguishes the "real" risk map updating function $P(\text{occ}|\mathbf{d}_k)$ (which is nothing but a mathematical concept) from the one we constructed from our sensor models, that we will use to update the risk map through updating law (4).

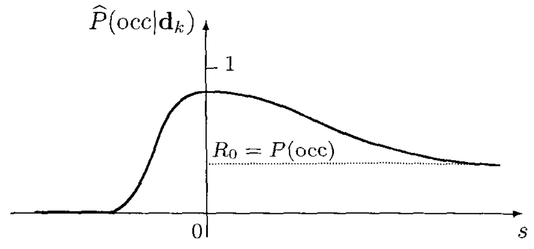

Figure 7: Shape of risk map updating function.

growing values of s (meaning that the grid point is behind an obstacle) then $\widehat{P}(\text{occ}|\mathbf{d}_k)$ tends to the *a priori* probability of occupancy $P(\text{occ})$, which makes sense since we cannot have any information on the occupancy state of the cell, that is "hided" by an obstacle (note that if $P(\text{occ}|\mathbf{d}_k) = P(\text{occ})$ then formula (4) yields $R_k = R_{k-1}$). The width of function (7) depends on both ξ (depth of objects) and σ^2 (uncertainty in measurements).

4.3 Local Path Planning using Dynamic Programming

Using function (5) and formula (4) the Tactical Planner is able to build and update the risk map. We will now briefly illustrate a novel algorithm [11] that, given the risk map, provides a sequence of grid points (i.e. a discrete path) connecting the two waypoints previously given by the Strategic Planner in a way that detected obstacles are avoided and path length is minimized; such algorithm is based on Dynamic Programming techniques [3].

We associate a *state* to each grid point (i.e. state space $\mathcal{S}$ coincides with grid $\mathcal{G}$), and we assume that from any state we may move, in one step, to any of its 26 neighbors in the three-dimensional grid; let $\mathcal{U}$ be the 26-element control space. We now associate a *cost* function $c : \mathcal{S}^2 \times \mathcal{U} \to \mathbb{R} : (s_i, s_j; u) \mapsto c(s_i, s_j; u)$ to each state pair (s_i, s_j) (i.e. to each state transition) and control u.

We will minimize, with respect to all possible control policies $g : \mathcal{S} \to \mathcal{U}$, the *total expected cost*:

$$J^g(s) := \lim_{N\to\infty} E\left[\sum_{k=0}^{N-1} \alpha^k\, c\Big(s(k), s(k+1); g\big(s(k)\big)\Big) \,\Big|\, s(0) = s\right],$$

where s is the starting state and $0 < \alpha < 1$; the *optimal policy* is given by the unique solution to Bellman's equation [3].

For our specific path-generation problem we defined cost as a linear combination of three (or more) terms. The first term is proportional to the value the risk map assumes in state s_i (if such value is higher than a certain threshold, say 0.8, then cost is set to a very high value, or infinity); this way trajectories that avoid obstacles will have a lower cost. The second term is proportional to the length of the path connecting states s_i and s_j, so that globally shorter paths will have a lower cost than others. The third term associates a lower costs to states at a certain altitude from ground, so that the agent is pushed to fly at those altitudes rather than at more costly ones in order to achieve a lower total cost. Finally, we could assign a *gain* (a negative cost) to those areas where the risk map assumes values that are close to 0.5 (maximum entropy), i.e. unknown areas; thus the agent would be *attracted* towards unexplored areas —such exploration might yield useful information about obstacle presence (or absence), and allow the Tactical Planner to generate a "better" path.[2] In fact, cost is the translation into mathematical terms of the *task* we want our agent to perform; for example, if we wanted our agent to reach its destination along a known trajectory (and avoid obstacles at the same time) we would just need to add to our cost function a term that is proportional to the distance between each state and the fixed trajectory.

The theory of Dynamic Programming provides fast and efficient techniques for finding approximate solutions to Bellman's equation (which is nonlinear), such as the *value iteration* and the *policy iteration* methods, which we successfully applied to our specific problem. Through computer simulation, we were able to obtain (in real-time) discrete trajectories connecting the first waypoint to the second one, that avoided obstacles and, at the same time, minimized global path length.

5 Conclusions & Future Work

This paper reflects a new attempt to address the problem of vision based autonomous navigation in a partially known environment. Offline computation exploits the a-priori knowledge about the environment, providing an initial guess about the optimal route. Online computation exploits the information provided by the vision sensor, capable of sensing the environment. The choice of a probabilistic sensor model and, as a consequence, of a probabilistic online path planning scheme is, according to the authors, the most appropriate to capture the natural uncertainty typical of every sensing process.

[2]This situation is referred to as *exploration-exploitation tradeoff*, where the first term refers to exploitation of current information while some exploration could increase such knowledge in order to plan a better path.

Multiresolution via wavelet transform allows to localize the level of accuracy required to minimize collision probability, making the approach scalable with respect to the size of the map. Future work will include further simulation; implementation and testing of the navigation system on an autonomous helicopter within the BEAR project at UC Berkeley [1].

Acknowledgment

Research is supported by grant (ONR)N00014-97-1-0946.

References

[1] http://robotics.eecs.berkeley.edu/bear.

[2] http://www.usgs.gov.

[3] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, Massachusetts, 1996.

[4] Ingemar J. Cox and John J. Leonard. Modeling a dynamic environment using a bayesian multiple hypothesis approach. *Artificial Intelligence*, 66(2):311–344, 1994.

[5] L.M. Reissell D.K. Pai. Multiresolution rough terrain motion planning. *IEEE Transactions on robotics and automation*, 14(1):19–33, February, 1998.

[6] H. Shim, H. J. Kim, S. Sastry. Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles. In *AIAA Guidance, Navigation and Control Conference*, Denver, Co, USA, 2000.

[7] H. Shim, T. J. Koo, F. Hoffmann, and S. Sastry. A comprehensive study of control design for an autonomous helicopter. In *Proccedings of the 37th IEEE Conference on Decision and Control*, Tampa, Florida, December 1998.

[8] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185–203, August 1981.

[9] Bruce D. Lucas and Takeo Kanade. An iterative registration technique with an application to stereo vision. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, volume II, pages 674–679, Vancouver, B. C., Canada, August 1981.

[10] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter-based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–238, 1989.

[11] Mario Micheli. *A probabilistic approach to three-dimensional autonomous navigation.* Laurea Thesis, University of Padova, Italy, 1999. In English, with a preface in Italian. Italian title: *Approccio probabilistico alla navigazione autonoma in tre dimensioni.*

[12] Hans Moravec and D. W. Cho. A bayesian method for certainty grids. In *AAAI 1989 Spring Symposium Series, Symposium on Mobile Robots*, 1989.

[13] L.-M. Reissell. Multiresolution geometric algorithms using wavelets:representation for parametric curves and surfaces. Technical Report TR 93-17, University of British Columbia, 1993.

[14] T. J. Koo, F. Hoffmann, H. Shim, B. Sinopoli, and S. S. Sastry. Hybrid control of model helicopter. In *Proceedings of IFAC Workshop on Motion Control*, Grenoble, France, October 1998.

[15] Sebastian Thrun. Learning metric-topological maps for indoor mobile navigation. *Artificial Intelligence*, 99(1):21–71, February 1998.