Rough Terrain Perception through Geometric Entities for Robot Navigation

Roberto Valencia-Murillo, Nancy Arana-Daniel, Carlos López-Franco and Alma Y. Alanís
Department of Computer Science
Universidad de Guadalajara
Guadalajara, Jalisco, México
Email: vamyur, n.arana.daniel, clzfranco, almayalanis@gmail.com

***Abstract*— This paper presents the implementation of a non-linear geometric cost function to be used with a learning to search algorithm (LEARCH) to robot navigation in rough terrains. The non-linear function introduced is a neural network trained with geometric entities as inputs (points, lines, spheres, planes). These inputs were codified using the Conformal Geometric Algebra framework in order to describe the features of the rough environment where the robot is going to navigate. The geometric entities contain implicitly more information about rough terrain that simple features obtained with image edge-detectors, furthermore by using them as descriptors, the dimension of the feature space is greatly reduced with regard to the dimension of features obtained with sophisticated feature detectors as SIFT or SURF. The advantages of using geometric entities with LEARCH algorithm are shown in the experimental results section of this paper.**

Keywords-Autonomous Navigation; Mobile Robots; Conformal Geometric Algebra; Learning From Demonstration

I. INTRODUCTION

Autonomous navigation in unstructured terrain for mobile robots represents a challenge to the robotic community. Performance of the robot traversing complex terrain depends on its perceptual and planning systems. The tasks of representing the environment and determining the robot actions based on this representation relay in individual performance and coupling of the mentioned systems.

The perceptual system creates a discrete model of the environment to be used by the planning system. In the simplest case, the interpretation determines which locations of the environment model the robot can or cannot traverse. However, in complex unstructured terrain this task is not that simple, the problem comes when the robot has to decide in nondeterministic way: sometimes objects may not be obviously traversable such as steep slopes, ditches, smaller (surmountable) objects, and widely varying vegetation. [1]

Different approaches address this problem; all of them implement a cost function to continuously couple the perceptual and planning systems. A cost function maps terrain features produced by perception into a scalar cost value with lower cost terrain being preferred over higher cost terrain. A planning system then computes a trajectory that minimizes the accrued cost of traversed terrain.

The most common approach to construct the cost function is engineering and manual hand tuning [1]; this is done with little or no formalism. However, there are some reusable frameworks for manual design and tuning of cost functions. This approach is time consuming; since complex environment necessitate full featured and high dimensional descriptions.

Physical simulation to attempt to predict the consequences of a robot traversing a patch of terrain is another common approach to engineering the problem [2], [3], [4]. Instead of requiring a mapping from perceptual features into cost, the robot maps from predicted states to costs. Another possible method is computing the probability of interaction with specific terrain patch which result on vehicle failure. These two methods present significant disadvantages. The first still needs construct a mapping from a description of a behavior to a cost function, and the second focus in the safety of the robot regardless the performance. Besides, it is dangerous because the robot needs to interact with non-traversable terrain to learn.

A different approach is supervised classification [1]. This technique reduces a high dimensional feature space into a lower dimensional space with more semantic meaning. It can be used to label terrain as traversable and non-traversable. However, even if classification is accurate, it rarely maps directly to the correct behavior.

Self-supervised learning from experience does not require expert interaction as other learning approaches. Instead, the robot uses its own interactions to learn how to interpret what it perceives. This learning can be used for predicting various terrain properties such as roughness [5], vehicle slip [6], soil cohesion [7], or vegetation height [8]. As other techniques based on the robot traversing capability, it ignores the possibility of relative preferences amongst equally traversable patches of terrain and is dangerous for the robot.

Given the difficulties presented by the manually engineering method to couple the perception and planning systems, an alternative solution consists of avoiding this problem by directly learning to map perception into actions. This can be accomplished through learning from demonstration. Although, a desirable behavior is very difficult to quantify, a human expert usually knows the actual correct behavior. So, in this approach, a human expert demonstrates the desire behavior directly to the robot, instead of tuning the cost function, and therefore the robot can tune itself to match the demonstration. [1]

To use a searching metric politic as the one used by the Inverse Optimal Control methodology is an alternative to action prediction. While optimal control seeks a trajectory

through a state space that optimizes some known metrics, inverse optimal control seeks a metric such that a known trajectory through a state space is optimal under that metric. Within mobile robotics, one similar approach would be to learn a cost function such that a robot planning system will reproduce expert demonstrated behavior.

Inverse Reinforcement Learning [9] was the first application of this idea to the Markov Decision Process commonly used for the motion planning in mobile robotics. This framework was later modified into a new approach known as Apprenticeship Learning, and it produced linear cost functions. However, there was no mechanism for explicit matching expert behavior.

The Maximum Margin Planning [10] framework addressed these problems by producing a deterministic solution while also ensuring an upper bound on the mismatch between demonstrated and planned behavior. This framework has been extended to non-linear cost functions with the learning to search algorithm (LEARCH) [11], [12].

In the LEARCH algorithm non-linear cost function might be used, such as parametric functions, neuronal networks, decision trees, etc. The results of the algorithm are affected by the selection of the non-linear cost function. In this paper neuronal networks were preferred over other non-linear cost functions for its capabilities of learning and generalizing. Also, the feature space in this paper is represented in Conformal Geometric Algebra, this means that the features of the environment are representing by geometric entities (points, lines, planes and spheres). This kind of features offers richer information of the environment where the mobile robot navigates and therefore their use increases the knowledge of the environment that the LEARCH algorithm can use to find a more accurate mapping between perceptions and actions as it is shown in Sec. V of this paper.

This paper address the complex unstructured terrain navigation with a learning from demonstration approach using neuronal networks trained with geometric entities as environment descriptors in order to improve the results of the LEARCH algorithm.

II. Learning From Demonstration

The objective of the learning from demonstration approach is to imitate the behavior of a human expert. The Maximum Margin Planning (MMP) approach constructs a cost function where states with lower cost represent the actions that the human expert would do.

In MMP there is a state space S which a planner operates. A feature space F is defined over S. That is, for every $x \in S$, it exists a corresponding feature vector $F_x \in F$. F_x represents the output of a perception system to be used by a planner, it must be mapped to a scalar cost value. Therefore, C is defined as the weighted sums of functions $R_i \in R$, where R is a space of limited complexity that map from the feature space to a scalar.

The cost of a state x is $C(F_x)$. Finally, a path P is defined as a sequence of states in S that lead from start s_e to goal g_e. The cost of an entire path is simply defined as the sum of the costs of all states along the path, or alternatively the cost of cumulative feature counts.

$$C(P) = \sum_{x \in P} C(F_x) \tag{1}$$

Consider a path P_e from a start space s_e to a goal g_e. Is reasonable to consider applying inverse optimal control if the example path is provided via expert demonstration, then; that is, seeking to find a cost function C such that P_e is the optimal path from s_e to g_e. If a regularization term is also added to encourage simple solutions, the task of finding an acceptable cost function from an example can be phrased as the following constrained optimization problem:

$$\begin{aligned} &\min O[C] = \mathrm{REG}(C) \\ &\text{subject to constraints} \\ &\sum_{x \in P_*} (C(F_x)) \geq \sum_{x \in P_e} (C(F_x)) \\ &\forall \hat{P} s.t. \hat{P} \neq P_e, \hat{s}=s_e, \hat{g}=g_e \end{aligned} \tag{2}$$

This optimization has a trivial solution $C(F_x) = 0$, this issue is addressed including a margin in each constraint. The size of the margin is dependent on the similarity between paths; this is encoded by a loss function $L(P_e, P)$ or L_e, because the loss function can be defined over a full path or over a single state.

$$L_e = \begin{cases} 1 \text{ if } x \in P_e \\ 0 \text{ otherwise} \end{cases} \tag{3}$$

The definition of the loss function is somewhat application dependant; the simplest form would be to simply consider how many states the two paths share (a Hamming loss). The effect of the scale of the margin is removed by the regularization term. The constrained optimization can now be rewritten as:

$$\begin{aligned} &\min O[C] = \mathrm{REG}(C) \\ &\text{subject to constraints} \\ &\sum_{x \in P_*} (C(F_x) - L_e(x)) \geq \sum_{x \in P_e} (C(F_x)) \\ &\forall \hat{P} s.t. \hat{P} \neq p_e, \hat{s}=s_e, \hat{g}=g_e \end{aligned} \tag{4}$$

Depending on the state space and the distance from s_e to g_e there is an unfeasible number of constraint. However, it is not necessary to enforce every constraint. For any candidate cost function, there is a minimum cost path between any two waypoints P_*. If this constraint is satisfied, by definition all the other will be satisfied. The constraint is now defined as:

$$P_* = \arg\min_{\hat{P}} \sum_{x \in \hat{P}} \left(C(F_x) - L_e(x) \right) \tag{5}$$

It may not be always possible to exactly meet this constraint therefore a slack term ζ is added and accounts for the error in meeting the constraint, and λ balances the tradeoff in the objective between regularization and meeting the constraint.

$$\min O[C] = \lambda \mathrm{REG}(C) + \zeta \tag{6}$$

However, the slack variable will always be tight. It will be always exactly equal to the difference in path costs. Therefore, it can be replaced in the objective function by the

constraint, resulting in the unconstrained optimization problem.

$$\min O[C] = \lambda REG(C) + \sum_{x \in P_e} C(F_x) - \min\left[\sum_{x \in P} \left(C(F_x) - L_e(x)\right)\right] \quad (7)$$

The final optimization seeks to minimize the difference in cost between the example path P_e and the (loss augmented) optimal path P_*, subject to regularization. [1]

$O[C]$ is convex, but non-differentiable; therefore, instead of gradient descent, it can be minimized using the sub-gradient method. Consider the sub-gradient in the space of cost functions.

$$\nabla O_F[C] = \lambda \nabla REG_F[C] + \sum_{x \in P_e} \delta_F(F_x) - \sum_{x \in P_*} \delta_F(F_x) \quad (8)$$

Where δ is the Dirac delta at the point of evaluation.

Applying gradient descent directly in this space would result in an extreme form of overfitting; essentially, it would involve rising or lowering the cost associated with specific values of F encountered on either path, and would therefore produce no generalization whatsoever. Instead, a different space of cost functions is considered

$$C = \left\{ C \middle| C = \sum_i \eta_i R_i(F), R_i \in \mathrm{Q}, \eta_i \in \mathbb{R} \right\}$$
$$\mathrm{Q} = \left\{ R \middle| R : F \to \mathbb{R} \wedge REG(R) < v \right\} \quad (9)$$

C is now defined as the space of weighted sums of functions $R_i \in$ Q, where Q is a space of functions of limited complexity, it maps from the feature space to a scalar. Choices of Q include linear functions, parametric functions, neural networks, decision trees, etc. [1]

A gradient descent update takes the form of projecting the functional gradient onto the direction set by finding the element $R_i \in$ Q that maximizes the inner product $\langle -\nabla O_F[C], R_* \rangle$. The maximization of the inner product between the functional gradient and the hypothesis space can be understood as a learning problem.

$$R_* = \arg\max_R \left\langle -\nabla O_F[C], R_* \right\rangle = \arg\max_R \sum_{x \in P_e \cap P_*} \alpha_x y_x R(F_x)$$
$$\alpha_x = \left|\nabla O_{F_x}[C]\right| y_x = -\mathrm{sgn}\left(\nabla O_{F_x}[C]\right) \quad (10)$$

In this form, it can be seen that finding the projection of the functional gradient involves solving a weighted classification problem, and defining Q as a class of regressors adds an additional regularization to each R_*. [1] [11] Intuitively, the regression targets, y_x, are positive in regions of the feature space that the planned path visits more than the example path and negative in regions that example path visit more than the planned path.

This approach can be understood as trying to minimizise the error in visitation counts. U is defined as the cumulative count of states $x \in P$ such that $F_x = F$:

$$U_+(F) = \sum_{x \in P_*} \delta_F(F_x)$$
$$U_-(F) = \sum_{x \in P_e} \delta_F(F_x)$$
$$U(F) = U_+(F) - U_-(F) = \sum_{x \in P_*} \delta_F(F_x) - \sum_{x \in P_e} \delta_F(F_x) \quad (11)$$

With this new definition of the problem, U can be used as target to the inputs of neuronal network. The states visited by the example path are separeted from the planned path. With this classification cost function maps the features from the example path with a lower cost than any other planned path.

III. Conformal Geometric Algebra

A. Geometric Algebra

Let G_n denote the geometric algebra of n dimensions, which is a graded linear space. G_n is defined by $G_{\{p, q, r\}}$, where p, q, and r stand for the number of basis vectors that square to 1, -1, and 0, respectively, and fulfill $n = p + q + r$. e_i is used to denote the basis vector i

The inner product of two vectors is the standard *scalar* or *dot* product, which produces a scalar. The outer or wedge product of two vectors is a new quantity, which we call a *bivector*. Thus, $b \wedge a$ will have the opposite orientation, making the wedge product anticommutative. The outer product is immediately generalizable to higher dimensions.

In geometric algebra, $G_{\{p, q, r\}}$, the geometric product of two basis vectors, is defined as

$$e_i e_j = \begin{cases} 1 \text{ for } i = j \in 1,\ldots,p \\ -1 \text{ for } i = j \in p+1,\ldots,p+q \\ 0 \text{ for } i = j \in p+q+1,\ldots,p+q+r \\ e_i \wedge e_j \text{ for } i \neq j \end{cases}$$

This leads to a basis to the entire algebra

$$\{1\}, \{e_i\}, \{e_i \wedge e_j\}, \{e_i \wedge e_j \wedge e_k\}, \ldots, \{e_1 \wedge e_2 \wedge \ldots \wedge e_n\} \quad (12)$$

Any multivector can be expressed in terms of this basis. For example $G_{\{4, 1, 0\}}$ has the basis:

$$\{1\}, \{e_1,\ldots,e_5\}, \{e_{12}, e_{13},\ldots,e_{45}\}, \{e_{123},\ldots,e_{345}\}, \{e_{1234},\ldots,e_{2345}\}, \{e_{12345} = I\} \quad (13)$$

B. Conformal Geometric Algebra

Geometric algebra $G_{\{4, 1\}}$ can be used to represent Euclidean vector space $\mathbb{R}^3$ in $\mathbb{R}^{4,1}$ [13]. This space has an orthonormal vector basis given by e_i and $e_{ij}=e_i \wedge e_j$ are bivectorial bases and a bivector e_{23}, e_{31} and e_{12} which together with 1 correspond to Hamilton's quaternions.

The unit Euclidean pseudo-scalar $I_e := e_1 \wedge e_2 \wedge e_3$, a pseudo-scalar $I = I_e E$, and the bivector $E := e_4 \wedge e_5 = e_4 e_5$ are used for computing Euclidean and conformal duals of multivectors.

1) *The Point:* The vector x_e representing a point after a conformal mapping is rewritten as

$$x_c = x_e + \frac{1}{2} x_e^2 e_\infty + e_0 \quad (14)$$

where the null vectors are the point at infinity $e_\infty = e_4 + e_5$ and the origin point $e_0 = {}^1/_2(e_4 - e_5)$ with the properties $e_\infty^2 = e_0^2 = 0$ and $e_\infty \in e_0 = 1$.

Given two conformal points x_c and y_c, we can define

$$x_c - y_c = (y_c \wedge x_x)\cdot e_\infty \tag{15}$$

and, consequently, the following equality:

$$(x_c \wedge y_c + y_c \wedge z_c)\cdot e_\infty = (x_c \wedge z_c)\cdot e_\infty \tag{16}$$

is fulfilled as well.

2) *Spheres and Planes:* The equation of a sphere of raduis p centered at point $p_e \in \mathbb{R}^3$ can be written as $(x_e \in p_e)^2 = p^2$. Since $x_c \in e_0 = {}^1/_2(x_e - y_e)^2$, where x_e and y_e are the Euclidean components, and $x_c \in e_0 = {}^1/_2\, p^2$, we can write the formula above in terms of homogeneous coordinates. Since $x_c \in e_\infty = -1$, we can factor the expression above to

$$x_c \cdot \left(p_c - \frac{1}{2} p^2 e_\infty \right) = \tag{17}$$

This equation correspond to the so-called inner product null space (IPNS) representation, which finally yields the simplified equation for the sphere as $s = p_c - {}^1/_2\, p^2 e_\infty$. Note from this equation that a point is just a sphere with a zero radius. Alternatively, the dual of the sphere is represented as a 4-vector $s^* = sI$. The advantage of the dual form is that the sphere can be directly computed from four points as

$$s^* = x_{c_1} \wedge x_{c_2} \wedge x_{c_3} \wedge x_{c_4} \tag{18}$$

If one of these points is replaced for the point at infinity, we get the equation of a 3D plane:

$$\pi^* = x_{c_1} \wedge x_{c_2} \wedge x_{c_3} \wedge x_{c_\infty} \tag{19}$$

And π in standard IPNS form

$$\pi = I\pi^* = n + de_\infty \tag{20}$$

Where n is the normal vector and d represents the Hesse distance for the 3D space.

Table 1 summarizes the entities defined in the conformal geometric algebra framework in its OPNS and IPNS representations.

IV. Learning from Demonstration using Neural Networks and Conformal Geometric Algebra

In the present work a feature space is declared in $G_{4,1}$, this means that geometric entities are going to be used in order to represent the features of the environment where the robot navigates. The geometric entities used to model the features in the environment are spheres, planes and points; they model three types of obstacles: vegetation, rubble and slopes.

LEARCH algorithm, as it was shown, is independent of the feature space. Using this advantage, we can use geometric entities or any other type of feature descriptors to construct the cost function in this work. We compare the efficiency of the LEARCH algorithm when it is using a hybrid feature space and a geometric feature space to describe the features of the environment, this was done in order to show the learning improvement achieved when the last one is used. In previous works [1], raw sensor data from the perception system (LiDAR scanner and cameras) is used to on-line train the cost function. After collecting sensor data, it has to be processed in order to obtain the values of the each feature vector, in [1] these values correspond to height, density and solidness of the objects presented in each sensing patch of the terrain. In this work the cost function was trained off line using satellite maps and the training data in the form of expert example behaviors was gathered by having several real human walking trajectories on the terrains, the process of translate features of the terrain into feature vector values was made manually by a human expert.

Neuronal Networks are in most cases a good election when capabilities of generalization are needed. The goal of learning from demonstration is that mobile robots become able to efficiently navigate through environments where features may be significantly different from the environment where they were trained. Therefore Neuronal Networks are good tools to address the problem.

A. Feature spaces

Satellite maps of rough terrains were used (see top image of Fig. 1) in order to obtain the traversability costs of each patch of terrain. Each satellite map of dimension 200×200 *m* was discretized into 2D grids. Each cell of the grid represents a patch of 4×4*m* of rough terrain and each one of these patches are represented as one vector in different features spaces: one hybrid feature space (similar to the feature space used in [1]) and one geometric feature space.

The hybrid feature space H is a vector space of features with dimension $dim(H) = 3$. A vector $X_{hyb} \in H$ is a ordered set of real scalar values: the vegetation value represents its density, rubble is represented as a binary value (representing the presence or absence of rubble in the patch of the terrain), and steepness of the slope is represented by a scalar value.

The geometric feature space SG is a vector subspace of $G_{4,1}$. The maximum dimension of the geometric entities used to represent each patch of the real map is $dim(SG) = 5$ for example, one sphere $s \in SG$ $s = e_1 + 2e_2 + e_3 - 0.5e + e_0$. Although $dim(SG) > dim(H)$ the information about the terrain contained in each geometric entity is more descriptive than the information contained in the hybrid feature vector, as it is shown in Sec. V.

TABLE I. Representation of Conformal Geometric Entities

Entity	IPNS Representation	PNS Dual Representation
Sphere	$s = p - \frac{1}{2} p^2 e_\infty$	$s^* = x_1 \wedge x_2 \wedge x_3 \wedge x_4$
Point	$x_c = x_e + \frac{1}{2} x_e^2 e_\infty + e_0$	$x^* = s_1 \wedge s_2 \wedge s_3 \wedge s_4$
Line	$L = nI_e - e_\infty m I_e$	$L^* = x_1 \wedge x_2 \wedge e_\infty$
Plane	$\pi = n + de_\infty$	$\pi = x_1 \wedge x_2 \wedge x_3 \wedge x_\infty$
Circle	$z = s_1 \wedge s_2$	$z^* = x_1 \wedge x_2 \wedge x_3$
Point Pair	$P_p = s_1 \wedge s_2 \wedge s_3$	$P_p^* = x_1 \wedge x_2$

B. *The neuronal network*

In this work it was used a multilayer perceptron (MLP) trained with Levenberg-Marquardt (LM) is the neuronal network used in this work as a cost function. The selection of LM to train the MLP was made based on the already demonstrated speed and accuracy of this training method.

The space *U* obtained with (11) contains the target values for each one of the inputs (hybrid space *H* or geometric space *SG*) of the MLP. In *U* there are negative values, but in robotic navigation negative values are not desirable. Therefore a logistic function is used as transfer function for the MLP and negative values of *U* are clamped to 0.

V. RESULTS

Tests of these feature spaces show clearly that Learning from demonstration using Neuronal Networks and Conformal Geometric Algebra is more efficient than using a feature space of engineering hybrid data. As seen in Fig 1 this is evidence: The learned costs of the map obtained using the hybrid cost functions are shown in the middle image of Fig. 1, as the reader can see, these costs are less descriptive of the features of the rough terrain shown in the map of the top of the same figure, with respect to the costs learned by the LEARCH algorithm using the geometric cost function (bottom image of Fig. 1) which shown that LEARCH algorithm learns a better (i.e. more descriptive) generalization of the traversability costs of each patch of terrain. Same results were obtained in all the experiments conducted so far in this work (they are not shown because of the space limit of this paper).

The Fig 2 shows the costs mapped by each approach in using the same map.

The difference between planned paths and example paths was validated for each cost function; the number of states from planned paths different from the example path were compared. The planned paths with geometric cost function where nearly exactly to the example path in many of the tests. A graphic with the precision of the planner using geometric cost functions and hybrid cost functions can be seen in Fig 3.

VI. CONCLUSION

In this paper we address the task of interpreting perceptual data to be used for efficient autonomous navigation on unstructured terrain. The Conformal Geometric Algebra approach was tested and compare with the engineering approach with evident improved results. We have shown that CGA can be applied on improving the performance of the learning from demonstration algorithm. Also, the Geometric Entities used in the tests offered a richer description of the environment; this is shown in the accuracy of the geometric cost function compared to the cost function with the engineering features.

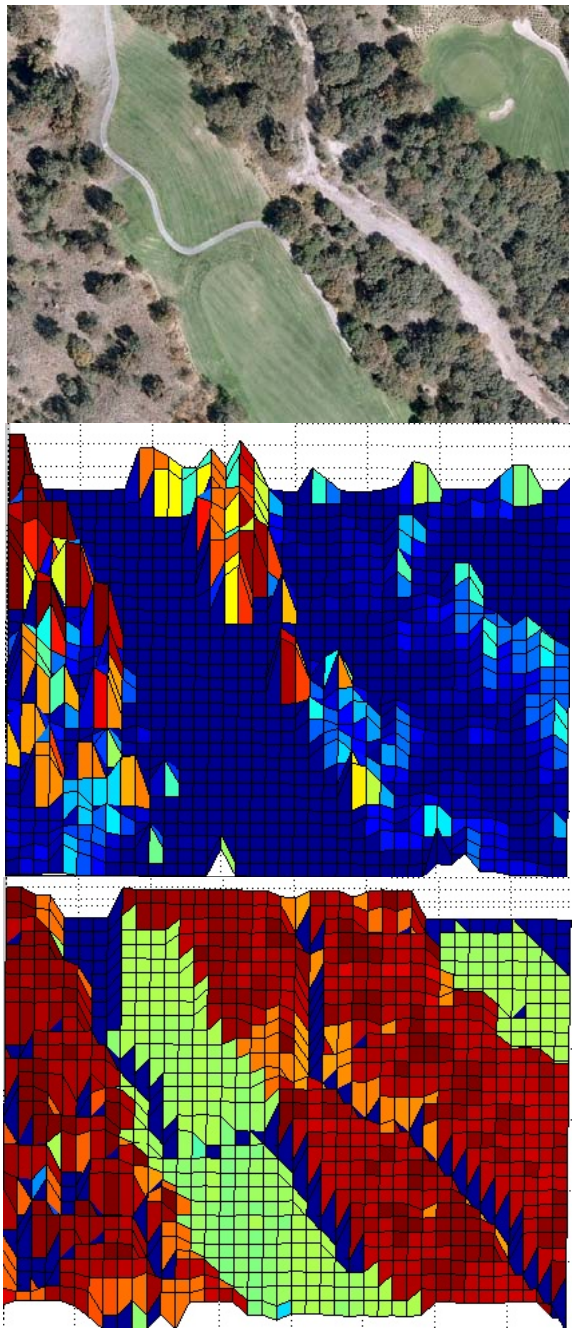

Figure 1. A test map for the algorithm. From the top to the bottom: Image of site, costs mapped with the hybrid cost function, and costs mapped with the geometric cost function. The reader can see that the description of the terrain obtained with the costs learned with the geometric function is much more detailed than the one obtained with the hybrid function.

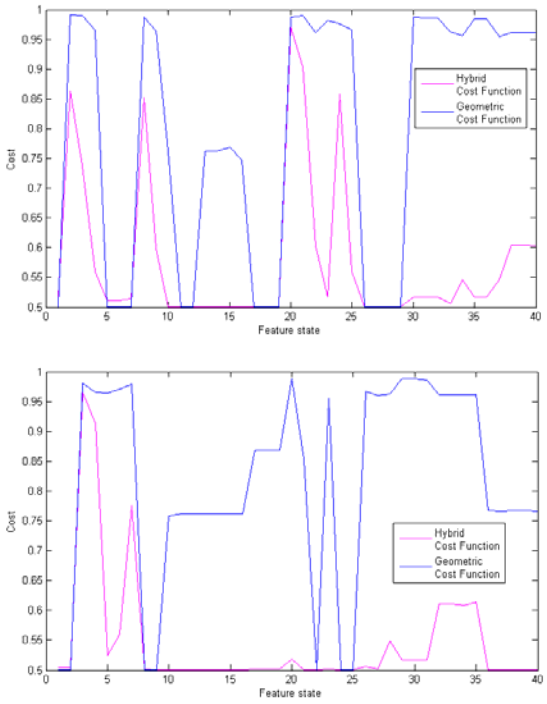

Figure 2. Differences of costs mapped with the two cost function over a patch of terrain. It can be seen that the geometric function generalizes costs of features of the environment which were not in the training set. This features are mapped as low cost and some cases as 0 by the hybrid cost function. Again, this shows that the level of detail of the rough terrain description is increased by using the geometric cost function (and it also matches better with the features of the real terrain).

Improper modeling of traversability preferences is harmful to robot performance as improper modeling of terrain preferences; it also contributes to the inability of some planning systems to properly recreate demonstrated behavior. In this work we were able of modeling proper traversability preferences using geometric entities as features and a neuronal network as a non-linear cost function to learn from an expert demonstration and how it was shown in the previous section the costs obtained using the geometric cost function allow us to claim that this function contains information which describes in a more detailed way the features of the rough environment without increase the dimensionality of the problem in a great way.

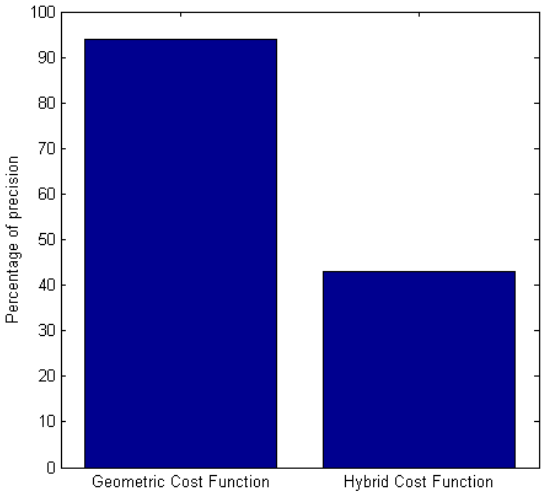

Figure 3. Precision of the planner using learned costs for 100 trajectories.

Future works will explore the implementation of other neuronal networks as cost functions along with the development of the automation of the process which translates features of the terrain into feature vector values.

ACKNOWLEDGMENT

This work has been partially supported by grants CONACYT CB-2008-01-106838, CB-156567 and CB-103191

REFERENCES

[1] David Silver and J. Andrew Bagnell and Anthony Stentz, "Learning from Demonstration for Autonomous Navigation in Complex Unstructured Terrain," The International Journal of Robotics Research, vol. 29, num. 12, pp. 1965-1592, 2010.

[2] Olin, K.E. and Tseng, D.Y., "Autonomous cross-country navigation: an integrated perception and planning system", IEEE Expert, vol. 6, num. 4, pp. 16-30, 1991.

[3] Iagnemma, K. and Genot, F. and Dubowsky, S., "Rapid physics-based rough-terrain rover planning with sensor and control uncertainty", Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, vol. 3, pp. 2286-2291, 1999.

[4] Helmick, Daniel and Angelova, Anelia and Matthies, Larry, "Terrain Adaptive Navigation for planetary rovers", Journal of Field Robotics, vol. 26, num. 4, pp. 391-410, 2009

[5] H. Dahlkamp and A. Kaehler and D. Stavens and S. Thrun and G. Bradski, "Self-supervised Monocular Road Detection in Desert Terrain", Proceedings of Robotics: Science and Systems, Philadelphia, USA, August 2006.

[6] Angelova, Anelia and Matthies, Larry and Helmick, Daniel and Perona, Pietro, "Learning and prediction of slip from visual information", Journal of Field Robotics, vol. 24, num. 3, pp. 205-231, 2007.

[7] Iagnemma, K. and Shinwoo Kang and Shibly, H. and Dubowsky, S., "Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers", Robotics, IEEE Transactions on, vol. 20, num. 5, pp. 921-927, 2004.

[8] Carl Wellington and Aaron Courville and Anthony (Tony) Stentz, "A Generative Model of Terrain for Autonomous Navigation in Vegetation", The International Journal of Robotics Research, vol. 25, num. 12, pp. 1287 – 1304, December 2006.

[9] Andrew Y. Ng. and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning", Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, pp. 663-670, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc.

[10] Nathan Ratliff and J. Andrew (Drew) Bagnell and Martin Zinkevich, "Maximum Margin Planning", International Conference on Machine Learning, July 2006.

[11] Nathan Ratliff and David Silver and J. Andrew (Drew) Bagnell, "Learning to search: Functional gradient techniques for imitation learning", Autonomous Robots, vol. 27, num. 1, pp. 25 – 53, July 2009, Springer.

[12] David Silver and J. Andrew (Drew) Bagnell and Anthony (Tony) Stentz, "High Performance Outdoor Navigation from Overhead Data using Imitation Learning", Robotics Science and Systems, June 2008.

[13] Li, Hongbo and Hestenes, David and Rockwood, Alyn, "Generalized homogeneous coordinates for computational geometry", Geometric computing with Clifford algebras, pp. 27 – 59, 2001, Springer-Verlag.