Learning from Previous Execution to Improve Route Planning

Johnathan Gohde * and **Mark Boddy** and **Hazel Shackleton**
Adventium Labs
111 3rd Ave S, Suite 100, Minneapolis MN 55401 USA
{firstname.lastname@adventiumlabs.com}

Abstract

Maintaining accurate maps for off-road route planning is an ongoing, error-prone, and time-intensive process. Missing or erroneous map information may result from glitches in translation of imagery data, from features not detectable in that data, or from changes in the environment that have occurred since the last update. These errors can lead to severely degraded planning performance, such as routes crossing areas that are in reality impassible or excessively hazardous, or routes that are much more costly in terms of time, fuel, or human effort than they need to be. In this paper, we describe G2I2, a map-based off-road route planner that learns corrections to the model through comparison of planned routes to the actual routes executed. Implemented using a field-tested off-road route planning package as the underlying planning engine, G2I2 modifies the performance of that engine by adjusting the input costs used by the planning algorithm. G2I2 is capable of learning both corrections to features in the current model (e.g., adjusting the cost associated with walking through waist-high grass), and corrections that encode features not present in the model at all, by modifying traversal costs based on geographic location.

1 Introduction

In this paper, we describe a specific approach to *iterative planning* in the domain of off-road route planning, in which the objective is to find a cost-minimal path from one point to another. In iterative planning we are concerned with finding a way to solve a succession of planning problems, improving the system's behavior over time.[1] For example, this improvement might come about through improved heuristics, leading to more effective search of the space of possible plans, or through corrections or additions to the domain model used in planning. In this work, we take the latter approach, modifying the domain model based on differences between plans generated using the existing model and "good" plans.

*This work was supported by DARPA, under Contract # D11PC20177

[1]As with many phrases of relatively recent coinage, "iterative planning" has several interpretations, even within the computer science community. The most recent articulation of the interpretation used here of which we are aware was in a AAAI 2012 Spotlight talk by David Smith (Smith 2012).

We have implemented our approach to iterative planning for generating off-road routes in a system called *G2I2*. In Section 2, we briefly discuss the route planning problem. Section 3 presents the current implementation of G2I2. Section 4 describes the learning model. The rest of the paper presents a set of experiments undertaken and summarizes the results obtained (Section 5), and discusses the implications of those results and the relationship of this work to other approaches to learning planning models (Section 7). Finally, we offer some concluding discussion in Section 7.

2 Route Planning

Path planning is an old and well-studied area of research. In this paper, we are specifically interested in path planning as applied to finding a way to travel from one physical location to another, generally out-of-doors. To distinguish this from other types of path planning such as maze solving, or moving physical objects through an occluded space (e.g., the piano movers' problem), we will refer to this as *route planning*.

Previous work has resulted in implemented systems that plan routes in spaces that correspond roughly to on-road and off-road scenarios. The former are most often graph-based planners. These kinds of planners are sufficiently well-understood to have been freely available via the Internet for at least the past decade, for example in Google maps. Off-road planners span a wider range, because there are significant qualitative differences between different types of terrain. Heuristic search using some form of remaining distance to the goal is a common technique. This works well in domains that are highly-obstructed, but not so obstructed as to be mazes.[2]

Route planning is a *model-based* process: it uses a *map* of the area, which may consist of a graph of streets, paths, corridors, and so forth, or as a description of the terrain on a pixel-by-pixel basis. For the work reported in this paper, the map is a given: it may contain numerous local errors, but the general structure of the area is accurately described. Due both to the presence of the map and to the availability of GPS information, localization is assumed: any route reported by a vehicle will be accurate within a reasonable error bound.

[2]It also works well in very open terrain, but that is less remarkable: so do much simpler techniques.

3 G2I2

Map-based route planning presents the classic difficulties faced by any model-based implementation. Maintaining accurate maps for off-road route planning is an ongoing, error-prone, and time-intensive process. Missing or erroneous map information may result from glitches in translation of imagery data, from features not detectable in that data, or from changes in the environment that have occurred since the last update. These errors can lead to severely degraded planning performance, such as routes crossing areas that are in reality impassible or excessively hazardous, or routes that are much more costly in terms of time, fuel, or human effort than they need to be. The terrain-based cost maps used for route planning share another common problem with model-based systems: there may be costs or map features that are very important to the human user, which would be prohibitively difficult to represent and keep current.

Ground Guidance ISK Integration (G2I2) addresses these issues by exploiting the complementary strengths of previous experience and knowledge-based planning. The presence of maps means that some form of plan can be generated even for areas that have never been previously traversed. Previous experience in the form of executed routes can be used to correct errors in those maps, and additionally to provide context not available in the maps at all, for example sensitivity to time-of-day or the prevailing weather patterns, both of which might significantly affect routing choices.

The user specifies a starting point and a destination, the intended mode of travel (on foot, or using any of a selection of vehicles), and a desired cost function, for example that the route be the fastest possible, or the most concealed, or any of several other criteria, some of which are only possible due to the information on previous execution maintained by G2I2. For example, one common criterion for military route planning in hostile environments is the desire not to use the same route too frequently or too predictably, so as to avoid some form of pre-positioned attack.

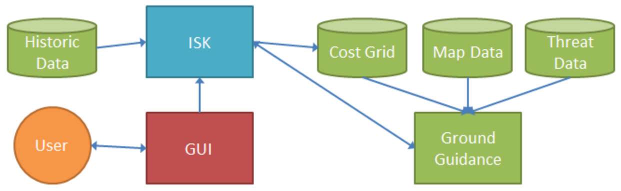

Figure 1: G2I2 functional architecture

The functional architecture for G2I2 is shown in Figure 1. G2I2 maintains a database of previous route executions, manipulating the domain model used by a commercial off-road route planning system called *Ground Guidance*. [3] Ground Guidance plans using a variable-resolution, modified A* search over an annotated map. This map is constructed using utilizing aerial photographs, land cover maps, digital elevation models (DEMs), and road data to plan optimal routes in mixed and urban terrain. Ground Guidance is called as a subroutine for every planning operation. These calls are made with varying cost maps and cost functions, producing differing results. Ground Guidance is additionally used to store and present map information.

[3] http://primordial.com/index.php/products/ground-guidance

4 Learning

G2I2 performs two learning tasks based on two sources of data. The first source of data is historic tracks. These tracks are presented to G2I2 as an unsupervised learning problem, used to build an initial set of preferences for features. This is very similar to the task addressed in (Silver, Bagnell, and Stentz 2008), where a training set of tracks assumed to be optimal (defined as the minimum summed cost) are used to induce a mapping from pixel labels in image data to cost values which are then used in planning. One difference is that we are dealing with a larger feature space, including time-varying meta-data associated with specific routes such as weather, as well as slope information that is not readily detectable in single overhead visual images.

The current implementation of G2I2 includes an elementary approach to inducing these costs, ignoring pixel features other than location. In other words, a given location is more attractive if some previous path has traversed it, rather than the more general approach where location features such as terrain type or slope are mapped to a traversal cost. We are currently in the process of generalizing this learning process, with specific attention to dimensionality reduction, motivated both by the large number of features (i.e., types of meta-data) associated with each pixel, and an intuition that the number of dimensions required for learning an effective mapping is much smaller.

The second learning task encompassed by G2I2 is the one which is the main thrust of this work. Based on the comparison of route plans *as planned* by G2I2, with the routes *as executed*, G2I2 further adjusts pixel traversal costs, both based on pixel meta-data, and based on location. The latter is significant because it allows the system to learn to avoid areas for reasons not represented by the map. Sometimes the human choice to traverse or avoid a given area will be either due to unmodeled features (the known presence of a specific threat, e.g.), or due to errors in the maps provided, such as a bridge that is no longer present.

This is a classic example of a supervised learning problem, and we address it as such. In this system, the routes are executed by humans. We assume the human route executors are taking the route they do because it is the best route for them, that is, the executed route has the lowest cost according to the route executor's cost function. If this route differs from the planned route, this is then because the planner calculated a cost for the planned route that was too low and the cost placed on the executed route by the planner is too high. We therefore alter the costs used by the route planner. This learning is performed online, improving the resulting generated plans with the feedback from each executed route.

The problem has some novel properties, including the presence of numerous, qualitatively very different types of features in the training instances presented. These are discussed in detail in our presentation of the application domain, in Section 2. We treat this as a parameter estimation problem, rather than as a pure classification problem. Instances (pairs of routes) are not used to infer a classification of routes, but to adjust an underlying set of costs used to compute a value for each instance. The result is a form of learning for planning in which what is being learned is not improved heuristics, but corrections to the underlying model. Previous work on similar kinds of learning includes *Learning by Demonstration* such as surveyed in (Argall et al. 2009), and *Maximum Margin Planning* (Ratliff, Bagnell, and Zinkevich 2006).

The assumption is that where the executed route diverges from the route as planned, that indicates some difference between the map representation of the territory and the territory itself. These differences can be separated into localized and non-localized phenomena. Localized phenomena are tied to a specific location, such as an obstruction along a path, a bridge out, or a location being mischaracterized in terms of terrain type or slope value. Non-localized phenomena include such things as incorrect costs on specific land cover types (perhaps driving a truck through waist-high grass is more costly than currently modeled) or slope preferences. In the results reported in this paper, we evaluate G2I2's ability to adjust its route planning through the modification of both localized and non-localized costs.

In G2I2, the form of learning being done is a modification of the cost of movement at specific locations on the map. This is not computing expected distance to a goal: the computed costs can be (and are) applied in planning numerous routes, for points located at different points on the map. This cost has several components, some purely geographic, but most related to features associated with the map as metadata, for example the kind of land cover or the slope at that point. These costs are modified as well by additional information associated with the plan itself, such as the mode of transportation to be used, and the time and weather during which the route will be executed.

Figure 2 shows the cost model used in G2I2. Dashed lines indicate features not presently calculable in Ground Guidance. Map, vehicle, and other data combine to form a feature vector. The combination of features is available to a variety of heuristic cost functions, each estimating the cost of traversing a given area based on the feature vector. These heuristic cost functions produce the component costs, such as speed or concealment, for the given vehicle traveling over the given map. These costs can be combined into a weighted sum, which provides the cost of movement at a point used by the search algorithm in Ground Guidance.

We represent the cost c_p of traversing a specific location p on the map as a local multiplier c_l times a sum of individual non-local feature costs $f_1 ... f_n$ with coefficients $c_1 ... c_n$ at that point:

$$c_p = c_l \sum_{i=1}^{n} c_i f_i \tag{1}$$

this cost function works for the current feature set, but will be revisited with additional features as noted in Section 7. The total cost of a route R is the sum of traversal costs for each location (pixel) along the route:

$$c_R = \sum_{i=R_{begin}}^{R_{end}} c_i \tag{2}$$

The form of learning performed by G2I2 is described in Algorithm 1, exploiting the notion that the route planner produced a route with too high a total cost. For both localized and non-localized costs, learning proceeds through a form of gradient descent.

Lines 2-8 find the proportions of features to be adjusted, using a minimum of 1% for any feature. This minimum is enforced to constrain the updates to reasonable values. Features present in the same proportions in each route are not adjusted. Features present in differing proportions are adjusted based on their relative proportion in the planned and executed routes. This proportion is stored in the map in line 6. Line 9 finds the value of a multiple for each feature that causes the executed route to have the same cost as the planned route when computed with the product of the initial feature cost and the feature proportion, with all other costs held constant. Using a multiple of the features' relative proportions as the basis of a cost update makes the planned route more expensive and the executed route less expensive.

When there is no cost update that renders the cost of the actual route lower than the planned route, such as is the case where the actual route traverses a longer path over a single, same feature found in the planned path, the local feature costs are updated instead. An update for the local feature costs is generated and returned, as shown in lines 10-14. Here the update doubles the cost of the local area where the routes differ.

When a cost update can be found, lines 15-17 store those updates, which are the feature proportions scaled by the multiplier found in line 9. These replace the initial, unscaled proportions originally stored on line 6. Line 18 totals the sum of the updates. If the total update to all features exceeds some threshold, indicating a radical departure from the prior costs, local feature costs are updated instead in lines 19-22. Choosing this threshold is an open issue, but the intention is to not allow a single pairing of routes to drastically alter the the feature costs. Line 23 returns a map of non-local feature cost updates or the local update weight if the threshold was exceeded.

Localized costs are adjusted through the addition of "cordons" that adjust the cost of movement for specific locations along routes. Due to the manner in which the underlying Ground Guidance planner operates, these localized costs are a multiplier upon the sum of non-local feature costs. Because localized differences necessarily exist when any difference in routes exists, these are treated specially. If no update to non-local costs can cause the planned route's cost to exceed that of the executed route, local costs are altered.[4] In the case that the updates are very large or cannot be found,

[4]In other words: if there is no explanation for the difference in routes in terms of the pixel meta-data, then we assume that there is some unmodeled feature of those particular locations that explains it.

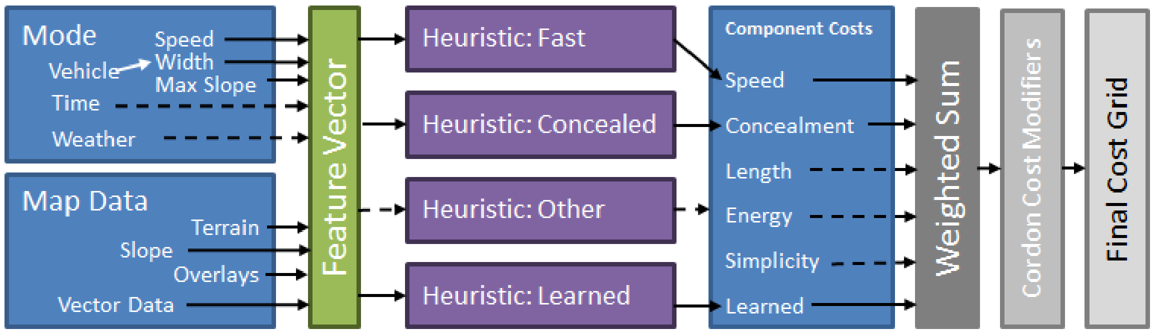

Figure 2: G2I2 Cost Model

Data: Planned route P, Executed route A
Result: Update map $M(f, u)$ mapping feature f with update u

```
begin
    for $f$ in union($P.features$, $A.features$) do
        $f_p \longleftarrow$ max(percentage of $P$ containing $f$, 1)
        $f_a \longleftarrow$ max(percentage of $A$ containing $f$, 1)
        if $f_p \neq f_a$ then
            $M(f) \longleftarrow f_p/f_a$
        end
    end
    Solve for $x > 0$ such that for each feature $f$ costs scaled by $xM(f)$ causes $cost_P = cost_A$ when calculated with the feature costs $cost_f xM(f)$.
    if For any feature, no such $x$ exists then
        $M.clear$
        $M(local) \longleftarrow 2$
        return $M$
    end
    for $f$ in $M$ do
        $M(f) \longleftarrow xM(f)$
    end
    $U \longleftarrow$ sum of all $u$ in $M(f, u)$
    if $threshold_{high} < U$ then
        $M.clear$
        $M(local) \longleftarrow 2$
    end
    return $M$
end
```

Algorithm 1: Calculate Cost Update

as would be the case when the non-local features are largely the same but the executed route takes a longer path, the cost along the divergent portion of the planned route is doubled.

Broadly speaking, segments of planned routes that are avoided have their costs incrementally increased, while the corresponding, divergent segments of the same route as executed will have their costs reduced. Using this form of cost adjustment, the system can accommodate not only erroneous information in the map such as the bridge example mentioned above, but *unmodeled* features. So long as those features are tied to particular locations, systematic attraction to or avoidance of those locations will over time result in plans that preferentially traverse or avoid those areas as well.

In the results reported in Section 5, the non-localized cost being updated is the land cover cost. As there are many types of traversable land cover, there are numerous features related to land cover in the feature vector for a single route (because the route may cross multiple land cover types). Because of limitations in the version of Ground Guidance used at the time of the experiments, to determine which costs in the land cover vector need to be updated, strict exclusion of land cover types between the two paths is used with fixed updates, opposed to relative percentages. If the planned path includes forest but no fields, while the actual path includes fields but no forest, then the cost to traverse fields will go down and the cost to traverse forests will go up.

Figure 3 shows an overlay of cost differences between base and learned costs to traverse different parts of the map, based on a learned correction to costs for different kinds of land cover. Areas of red indicate the learned cost is higher, while areas of green indicate lower cost. Color saturation indicates the relative difference in costs. White indicates no cost difference, and black is an area that is impassible in both cost models.

Other non-localized costs can be updated in the same way, adjusting the heuristic cost functions to yield an altered cost to areas that contain the non-local cost in the learned situation. If pairs of planned and actual paths indicate that, for example, paths over areas of lower slope are consistently taken over planned routes on higher slope, the costs of traversing high slope areas will be increased and the costs of traversing low slope decreased.

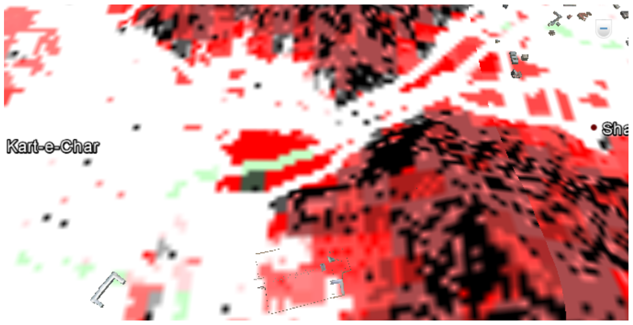

Figure 3: Terrain cost difference between base costs and learned costs

Terrain Type	Initial	Learned	Actual
Deciduous Forest	0%	5%	5%
Developed, High	2%	2%	2%
Secondary Road	100%	75%	75%
Trail	33%	30%	33%
Open, Barren	7%	33%	33%
Open, Grassland	7%	4%	5%
Open, Shrub	7%	25%	25%
Stream, Intermittent	7%	20%	10%
Stream, Shallow	7%	2%	2%
General Agriculture	7%	4%	4%

Table 1: Experiment 1 initial, learned, and actual costs (percentage of maximum speed) over varying terrain types

5 Experimentation

We report on three different experiments. In the first, the planner is initially presented with a map in which the costs of movement for various land cover types are set to incorrect values. More precisely, this "terrain cost" is a measure of how fast a given type of vehicle can travel over that type of land cover, as a percentage of the vehicle's maximum speed. The as-planned route is generated using this incorrect information. A corresponding as-executed route is generated using the real information.[5] Paired routes are generated sequentially, for randomly-chosen start and end points, up to 1.5 kilometers apart, within a 120 km^2 area.

These costs are non-localized in the sense described above: adjustments to terrain costs apply for that type of land cover anywhere it appears on the map, not just along this particular pair of routes. This type of learning will converge, if it converges, only on a corrected set of *relative* terrain costs. While there is information available on the total cost of both the planned and executed route in a given pair, this information is local rather than global: small differences in terrain costs may have disproportionate effects on route costs for specific routes. There may be additional errors in the learned costs for such things as terrain types that occur rarely on the map and so appear infrequently in route pairs. For this experiment, we defined "convergence" to have occurred when fewer than 5% of the last N iterations produced changes in any terrain costs. Since there are 10 different terrain costs that might individually be adjusted, this means either that most of the costs are not changing, or that some larger set of them are changing, but very infrequently. Using larger values of N results in fewer errors in the final learned terrain costs, but converge more slowly. For the results reported here, we used a value of 400.

As discussed previously, our ultimate objective is not corrected terrain costs, but better routes, meaning in this case routes that are closer to those generated using the real map data. We evaluate this by comparing route pairs using the final learned costs and the real data, measuring the distance between multiple points along the two routes in a given pair. This divergence is then averaged over a large number of route pairs.

The second experiment evaluates the ability of G2I2 to learn to generate improved plans in the presence of localized map errors. For this experiment, we chose an subsection of the map including a river and several bridges across it. One of these bridges is then rendered impassible in the "real" data (as an edit to what is, in fact, real map data for a part of Afghanistan), but not in the map provided for generation of as-planned routes. Route pairs are then generated as in the first experiment, for randomly-chosen start and end points on opposite sides of the river. Divergence between the planned and executed routes in each pair in turn is used to adjust the map cost. In this case, the cost being adjusted is spatial: the specific area traversed by each route is affected, rather than the cost associated with some form of meta-data applying to multiple map locations.

The third experiment was designed to show that G2I2 can learn in the presence of both localized and non-localized errors. This experiment recapitulated the first experiment described above, with the addition of the impassible bridge as in the second experiment. The objective in this case was to show that localized and non-localized errors can be addressed at least somewhat independently: the adjustment of non-localized costs will converge in the presence of localized errors, which can subsequently be dealt with as in the second experiment. All routes for all three experiments were generated using a Jeep as the mode of transport.

Table 1 shows results for the first experiment, which was run five times to convergence as described above. Listed are the changes to ten terrain features found in the area, of which eight were altered from the initial values for generating actual routes. The table shows the results of the single run that took the median time to converge, which was 4,000 iterations. Each of the five experiment runs took fewer than 6,000 iterations to converge.

Furthermore and as previously discussed, our primary interest is not in how accurately the system learns these costs, but in the degree to which planning improves. Table 2 compares planning results using initial and learned costs. In each case, the quality of the plans generated is evaluated by computing the "divergence" between the planned route and a route generated for the same endpoints using the real

[5] This cost information is "real" in a strong sense of that word: the map we used is drawn from actual terrain and cost data for an area in Afghanistan.

Initial Costs	Value
Maximum Divergence	99.83%
Average Divergence	38.53%
Routes with Divergence	724
Learned Costs	**Value**
Maximum Divergence	98.55%
Average Divergence	1.83%
Routes with Divergence	94

Table 2: Experiment 1 - Comparing the quality of routes generated using initial and learned costs

data. Divergence is calculated by dividing the length of the portion of the planned route that does not overlap the actual route by the length of the entire planned route.[6] Maximum and mean divergence is computed using 1000 pairs of routes between randomly generated start and end points.

The results summarized in Table 2 are very strongly positive. The number of routes for which there is any divergence (i.e., any difference between the planned route and the "real" one) drops dramatically, though it remains close to 10%. The more striking result is the average divergence. On average, less than 2% of the total extent of a given route differed from the desired route. Considering that at least one route out of 94 was essentially completely divergent, the average divergence for the other 93 routes was probably closer to 1%.

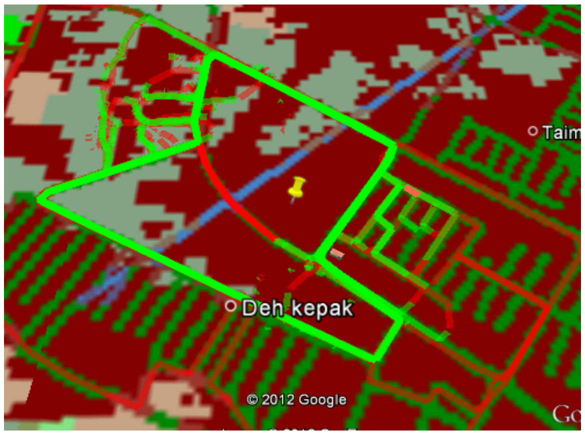

Figure 4: Local cost updates in Experiment 2

In the second set of experiments, 100 pairs of routes are planned from one side of the river to the other, using models that do and do not include the erroneously passable river crossing. When the planned route differs from the actual route, the portions that differ are overlaid with *cordons* that alter the local cost. Figure 4 shows the resulting set of cordons projected onto the map. The blue line through the center of the image is the river, dull red are urban areas, dull green are roads. Bright red areas, such as in the center crossing the river, are areas of increased cost. These coincide with the bridge that is out, as well as extending out along the road leading to the bridge. Bright green areas, such as those both north and south of the bridge that is out indicate reduced cost.

The first five plans generated using the erroneous map attempted to traverse the impassible bridge, resulting in a net increase of the local movement cost in that area by a factor of approximately 7.5. After those five, all other generated routes avoided that bridge. At least for relatively simple feature errors, it is clearly the case that small numbers of training examples and relatively minor local cost modifications can be effective at improving the routes generated.

This example is not entirely realistic. In the more likely scenario, the actual route would traverse the planned route towards the impassible bridge, diverging only when close enough to the bridge to detect the problem. In the "executed" routes generated by Ground Guidance using real data, this knowledge was a given, so the actual routes avoided any inefficiency. This is a minor issue, though: the actual route will still not cross the bridge, thus the cost of crossing will be increased. Once the cost of crossing the bridge has increased enough (after 5 attempts, in this experiment), then the planner will use the other bridges, planning minimal-cost routes to cross them, rather than heading for the original bridge. The end result will be almost exactly the same in terms of planner performance.

Our final experiment was intended to evaluate the degree to which localized feature errors interfere with learning corrections to non-localized costs. Similar to the second experiment, the real map was modified to mark several choke point areas on the map as impassible. After that, we proceeded to learn terrain costs as in the first experiment, but over an area of reduced size (10 km^2), which eliminated the "Grassland" and "Shallow Stream" terrain types.

Terrain Type	Initial	Learned	Actual
Deciduous Forest	0%	6%	5%
Developed, High	2%	2%	2%
Secondary Road	100%	80%	75%
Trail	33%	30%	33%
Open, Barren	7%	33%	33%
Open, Shrub	7%	25%	25%
Stream, Intermittent	7%	10%	10%
General Agriculture	7%	4%	4%

Table 3: Experiment 3 initial, learned, and actual speeds over varying terrain

Table 3 shows the initial, learned, and actual costs for this experiment. The experiment was run several times, with runs taking on average more iterations to converge than in the first experiment. Each run converged within 10,000 iterations. The table shows the results of the run that took the median time to converge, which was 8,000 iterations. Convergence in this experiment is slower than in the first experiment, probably due to the presence of the localized feature errors' introducing differences in between actual and learned costs that are not being adjusted in the learning process.

Table 4 compares planning results using initial and

[6]In other words, any difference between the planned and actual route, regardless of how close it may be, is treated as an error.

Initial Costs	Value
Maximum Divergence	99.98%
Average Divergence	39.23%
Routes with Divergence	722
Learned Costs	Value
Maximum Divergence	99.97%
Average Divergence	8.20%
Routes with Divergence	352

Table 4: Experiment 3 - Divergence of 1,000 paths planned with learned and actual results

learned costs, with divergence computed as before. Maximum and mean divergence is again computed in each case using 1000 routes between randomly generated start and end points. The improvement is still significant, but notably weaker than in the first experiment, probably because areas that are marked impassible in the real data do not have their costs adjusted in the learned map in this experiment.

6 Related Work

(Rogers, Fiechter, and Langley 1995) describes an on-road navigation system that models the user's preference for different classes of road, such as highway, freeway, arterial roads, and local roads, along with other route features such as driving time, distance, number of turns, and number of intersections. In this system, the user is presented with a proposed route, which can be accepted or rejected. Upon rejection, new routes are generated and the preference model updated based on the ultimate route selected compared to those rejected. Even beyond the restriction to on-road routes, this approach is strictly simpler than ours. For example, the system does not learn localized model changes. If the user knows that a proposed freeway is under construction, the rejection of routes including this freeway will update the preference for all freeways, not just the one rejected.

In (Letchner, Krumm, and Horvitz 2006), a route planner called TRIP is described that uses previously executed plans in the form of GPS tracks to inform future route generation. The previous trip information is used in two ways. First, it is used to update speed information along roads for the time at which the trip was recorded. Second, a user's inefficiencies are bundled into a preference factor for non-optimal routes. TRIP then plans over route segments, discounting previously-taken segments by the preference factor. This work is related to an earlier version of G2I2, which used only historical track information, rather than integrating that information back into an annotated map as in the current system.

There is a long history of research on learning planning models, including filling in incomplete domain models, for example (Gil 1992), and diagnosing and learning action definitions (Wang 1995).

Work specifically on learning to adjust a cost model for route planning includes the work by (Ratliff, Bagnell, and Zinkevich 2006) and (Silver, Bagnell, and Stentz 2008), discussed in Section 4. Work on *probabilistic roadmaps* such as (Kavraki et al. 1996) is superficially similar, but works in configuration space for holonomic robots, rather than terrain traversal. Finally, our work can be differentiated from previous work on map learning such as SLAM[7] in several ways. Notably, we start with a map, albeit one that may contain errors of various kinds, and localization is not part of the problem.

7 Discussion and Future Work

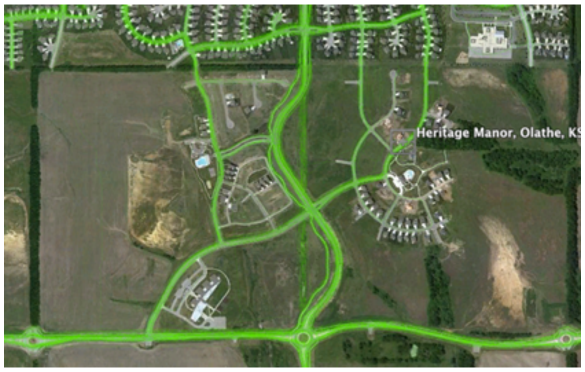

Figure 5: Older GPS tracks along a straight road.

We have presented G2I2 as an instance of "iterative planning," in which planning performance improves over time specifically because of the results of executing previous plans. There are other ways in which we can view plans as objects subject to manipulation, rather than the end result of the process. For example, in work left out of this paper for reasons of both space and focus, we have implemented a capability for generating multiple plans, either as a set of options roughly following a Pareto frontier in a multi-attribute value space, or in the generation of *interestingly different* plans against the same objective function.

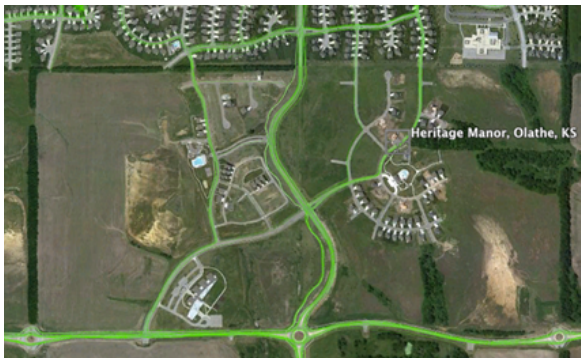

Figure 6: Tracks filtered by time, removing outdated paths from the map.

In this paper, we have shown that even a simple form of learning will lead to improved route planning performance over time, even in the presence of confounds such as unmodeled local errors. However, the work presented here

[7]Simultaneous Localization And Mapping

uses only a fraction of the available map features. We have shown nothing regarding costs associated with slope, or with computed meta-data such as "concealment." More significant and of more interest for future research is the use of meta-data associated with routes that is not directly associated with the map. In particular, there is a temporal dimension: the prevailing conditions *when* the route was executed are relevant, and provide an additional source of data for learning to improve planning.

A simple example of the use of temporal meta-data is illustrated in Figures 5 and 6, both showing a set of GPS tracks in Olathe, KS, gathered over a period of several months. In Figure 5, there is a straight, vertical track through the center of the map, showing the presence of tracks that took that route. Figure 6 shows the same area, with tracks filtered to exclude those before a specified date. In this figure, the vertical feature is missing. The explanation is visible in the satellite image on which the tracks are overlaid: There is a curving road through the area in question, which was only recently completed. Previously, the road ran straight north and south.

As the number of features increases, the difficulty of the learning problem increases rapidly. Correlations among features may provide a means of reducing this complexity.[8] *Principal Component Analysis* (PCA), and sparse variants such as DSPCA (d'Aspremont et al. 2007), may be used to reduce the dimensionality of the problem. These methods are limited to linear combination of variables, but can be extended to non-linear combinations through the use of kernel methods (Schlkopf, Smola, and Mller 1996). Another means of dealing with high dimensional problems would be to use support vector machines. Specifically, support vector regression machines (Drucker et al. 1997) may support finding a non-linear mapping of features to the underlying cost function. At this early stage, the kinds of correlations among features that may be required for effective dimensionality reduction are unknown, which is why such a wide range of techniques are potentially relevant.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robot. Auton. Syst.* 57(5):469–483.

d'Aspremont, A.; Ghaoui, L. E.; Jordan, M. I.; and Lanckriet, G. R. G. 2007. A direct formulation for sparse pca using semidefinite programming. *SIAM Review* 49(3):434–448.

Drucker, H.; Burges, C.; Kaufman, L.; Smola, A.; and Vapnik, V. 1997. Support vector regression machines. *Advances in neural information processing systems* 155–161.

Gil, Y. 1992. *Acquiring Domain Knowledge for Planning by Experimentation*. Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University.

Kavraki, L.; Svestka, P.; claude Latombe, J.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, 566–580.

Letchner, J.; Krumm, J.; and Horvitz, E. 2006. Trip router with individualized preferences (TRIP): incorporating personalization into route planning. In *Proceedings of the 18th conference on Innovative applications of artificial intelligence - Volume 2*, IAAI'06, 1795–1800. AAAI Press.

Ratliff, N.; Bagnell, J. A. D.; and Zinkevich, M. 2006. Maximum margin planning. In *International Conference on Machine Learning*.

Rogers, S.; Fiechter, C.-N.; and Langley, P. 1995. An adaptive interactive agent for route advice. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99)*, 198–205. ACM Press.

Schlkopf, B.; Smola, A.; and Mller, K.-R. 1996. Nonlinear component analysis as a kernel eigenvalue problem.

Silver, D.; Bagnell, J. A. D.; and Stentz, A. T. 2008. High performance outdoor navigation from overhead data using imitation learning. In *Robotics Science and Systems*.

Smith, D. 2012. Planning as an iterative process. In *Proceedings of AAAAI*.

Wang, X. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the 12th International Conference on Machine Learning*.

[8]For example, grassland is generally fairly flat.