Energy-efficient Path Planning for Solar-powered Mobile Robots*

Patrick A. Plonski
Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455
e-mail: plonski@cs.umn.edu
Pratap Tokekar
Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455
e-mail: tokekar@cs.umn.edu
Volkan Isler
Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455
e-mail: isler@cs.umn.edu

Received 9 August 2012; accepted 22 March 2013

We explore the problem of energy-efficient, time-constrained path planning of a solar-powered robot embedded in a terrestrial environment. Because of the effects of changing weather conditions, as well as sensing concerns in complex environments, a new method for solar power prediction is desirable. We present a method that uses Gaussian Process regression to build a solar map in a data-driven fashion. Using this map and an empirical model for energy consumption, we perform dynamic programming to find energy-minimal paths. We validate our map construction and path-planning algorithms with outdoor experiments, and we perform simulations on our solar maps to further determine the limits of our approach. Our results show that we can effectively construct a solar map using only a simple current measurement circuit and basic GPS localization, and this solar map can be used for energy-efficient navigation. This establishes informed solar harvesting as a viable option for extending system lifetime even in complex environments with low-cost commercial solar panels. © 2013 Wiley Periodicals, Inc.

1. INTRODUCTION

Mobile robots have the potential to perform many critical outdoor tasks, but their feasibility for long-term deployment is limited due to energy concerns. A possible method to increase the battery life of robots is by harvesting energy from the environment, e.g., with photovoltaic solar panels. Solar harvesting has proven to be useful in marine and extraterrestrial robotics applications (Carsten et al., 2007; Sauze and Neal, 2011) which take place in open space. However, in applications where the robot must operate in complex environments, such as urban search and environmental monitoring, the utility of solar harvesting is not obvious. In this work, we focus on extending the battery life of mobile robots using solar panels in such settings.

We study techniques for energy-minimizing path planning for a mobile robot with a photovoltaic panel that uses recent measurements of solar intensity as its only source of information about future solar power. This is an interesting problem because there are many applications where mobile robots do not necessarily have the sensors or computing power to estimate solar maps using sophisticated techniques such as raytracing on three-dimensional (3D) models of the environment. However, energy-efficient paths are still desired. Intuitively, it seems feasible for a good solar map of the environment to be built using only the recent solar measurements, if the robot is in generally the same region for long enough. Our approach is well-suited for applications such as environmental monitoring, data muling, and patrolling, in which a robot visits regions in the environment repeatedly (Dunbabin and Marques, 2012). As demonstrated in Section 5.4, our algorithm can be used as a subroutine for energy-efficient waypoint navigation in such applications.

The representative problem we address in this work is the following: suppose we have a mobile solar-powered robot that has been performing a task while also logging the power received from an onboard solar array. Each solar measurement is associated with an estimated robot position. Suppose the robot is required to perform a new task that requires it to reach a goal position within some time limit.

*An earlier version of this paper appeared in ISER (Plonski et al., 2013). That version had no background section, a significantly condensed solar modeling section, and it did not contain any of the June 9 solar maps or sections 5.2, 5.3, or 5.4.

How can the robot plan the path that minimizes its net energy consumption? We break this problem down into a mapping segment where we construct a solar map from the solar measurements the robot has happened to take so far (Sections 2 and 3), and a planning segment where we use the solar map in combination with a power-to-drive model to compute the best path to the goal (Section 4). We present results from experiments that demonstrate the utility of our techniques (Section 4.6). We also present simulation results on our solar maps to demonstrate the benefit gained by adding solar panels on a robotic platform, demonstrate the benefit gained by using our path planner over a naive path planner, examine the robustness of our algorithm to the constantly changing angle of the sun, and demonstrate a practical data mule application of our methods. (Section 5).

1.1. Related Work

Energy-efficient planning for mobile robots has received increased attention recently. The problem of modeling the power consumption of motion, sensing, communication, and embedded computing for commercially available robots was studied by Mei (2006). These power models were used to compare various strategies for high-level tasks such as coverage, exploration, and networking between robots, with an aim to increase the lifetime of the system.

Motion is a major source of power consumption for typical robots. The problem of minimizing the energy consumption by optimizing the velocity profiles for a given path was studied by Tokekar et al. (2011), Wang et al. (2011) and Kim and Kim (2007). Sun and Reif (2005) studied the problem of finding energy optimal paths between two points on terrains where the cost depends on friction and gravity and is thus direction-dependent. They presented an approximation algorithm for finding the minimum energy path, but they did not optimize the velocity profile along the path. Liu and Sun (2011) recently studied the problem of computing energy-efficient paths and trajectory profiles by optimizing the parameters of Bezier curves using an energy-based heuristic. However, the presented method is not guaranteed to minimize energy, and the general problem of simultaneously optimizing the path and velocity for a given start and goal pose remains unsolved.

Energy-efficient motion planning in the context of applications such as coverage is a subject of recent study. An example is Derenick et al. (2011), who studied the problem of maintaining persistent coverage using a network of robots by deriving control laws that allow robots with depleted batteries to reach corresponding access points. Similarly, Jensen et al. (2011) presented strategies for reconfiguring robot formations for a patrolling application in the case in which some robots run out of power and need to be replaced by fully charged robots.

Energy optimization for data mule applications has also been addressed recently. Sugihara and Gupta (2009) present path-planning algorithms for a data muling system with the goal of optimizing the trade-off between the energy consumption of the sensors and latency of the data carried by the robot. Recently, Bhadauria et al. (2011) studied the problem of finding time-efficient trajectories for a mobile robot downloading data from a set of wireless nodes, and by setting the parameters proportional to energy cost their approximation algorithm can minimize energy instead of time. In the above-mentioned work, the energy consumption of the robot, however, is not considered. In this paper, we present path-planning techniques that can potentially be useful for such applications.

The aforementioned works have not considered energy harvesting from the environment, and solar-aware path planning has received limited attention. In extraterrestrial applications and some environments on earth [e.g., in Antarctica (Ray et al., 2007)], collected solar energy can be treated as mostly independent of the path chosen. The TEMPEST mission-level path planner (Tompkins et al., 2006) uses ephemeris software to determine the position of the sun and then performs raytracing on known nearby terrain to build a solar map that is used to estimate the energy cost of paths. This is feasible when nearby terrain is known or when it can be accurately detected, but many otherwise feasible platforms for long-term environmental monitoring lack the necessary sensors to do this. In this paper, we focus on predicting solar power in complex environments using only the robot's position estimates and solar power measurements.

2. BACKGROUND

Before we present our algorithms, we first provide a brief overview of the factors that determine how much solar power a photovoltaic panel generates, and we discuss methods that have been previously used to model and predict solar power.

2.1. Photovoltaic Power Generation

The amount of current I a solar cell will output when it is fixed to a particular voltage V is the solution to the equation

$$I = I_L - I_s(e^{(V+IR_s)/V_T} - 1) - \frac{V + IR_s}{R_p},$$

where I_s is the reverse saturation current of the diode and $V_T = \frac{kT}{q}$, which is known as the thermal voltage (Lorenzo et al., 1994). I_L is proportional to the number of photons that impact the solar cell, and therefore so is I. I decreases with higher voltage, but the effect is not pronounced until the diode knee voltage is reached at around 0.5 V for a silicon cell. The knee voltage increases with decreased temperature, but in general the voltage limit varies much less than the current. Some systems use Maximum Point Power Trackers to adjust the voltage to the point where the cell puts out the most energy, and in these systems temperature is an important factor when power modeling. Other systems enforce a constant voltage; when this is the case, temperature can be neglected as long as there is a sufficient margin between the diode knee voltage and the induced voltage on the cell.

Because the voltage of an individual cell is low, cells are usually connected in one or more strings such that each string is electrically in series. These strings have the property that the amount of current output is limited by the *weakest* cell in the string (ignoring the effect of bypass diodes). The weakest cell could be the cell with the smallest dot product between its normal vector and the sun angle vector, or it could be a cell that happens to be in a shadow. We will see in Section 3 that this strong response to partial shading of the array complicates our task of map construction.

Sunlight reaches a solar panel in three different ways (Goswami et al., 1999): If it comes directly from exactly the part of the sky that contains the sun, it is called direct insolation. If it comes from any other part of the sky, it is called diffuse insolation. Finally, if it comes from anywhere else (i.e., from terrain or objects), it is called reflected insolation. Reflected insolation is most relevant when a solar panel is tilted toward a reflective surface (such as snow) or near a reflective building. On a sunny day direct insolation is high and diffuse insolation is low, whereas on a cloudy day direct insolation is low and diffuse insolation is high (and total insolation is much lower than on a sunny day). If a cell has no line of sight to the sun it is in a shadow, and direct insolation drops to zero. However, for diffuse insolation to drop to zero, the entire sky must be blocked. Therefore, we can expect shadows and thus the correct solar map to be much sharper on a sunny day than on a cloudy day.

2.2. Sunlight Modeling

If the amount of solar radiation incident on the panels is known, it is a simple matter to calibrate the aforementioned model to the specific system and arrive at a solar power estimate. But how can we estimate the amount of sun?

There is a large body of work on daily solar power prediction, mostly emphasizing static solar collector installations. The position of the sun relative to the earth at any time can be determined from well-known orbital mechanics equations [this was used for path planning in, for example, Shillcutt (2000)]. Threlkeld and Jordan (1958) showed that once the elevation angle is known, it is possible to estimate for a clear day the degree of attenuation the direct radiation experiences by traveling through the atmosphere (this attenuation is greater when the sun is lower in the sky because the radiation has to travel through more atmosphere) from an assumed optical depth k. They also showed that the diffuse insolation on a clear day, is proportional to the direct radiation, with the parameter C that relates them varying with dust and water vapor in the atmosphere. On a clear day, diffuse radiation does not come equally from all parts of the sky but instead comes more from the part of the sky close to the sun; Temps and Coulson (1977) showed how to calculate diffuse insolation to a tilted panel on a sunny day, and Klucher (1979) added a cloudiness parameter so that the model for diffuse insolation can smoothly vary between a cloudless day and a completely overcast day. On a completely overcast day, direct insolation can be assumed to be 0, but on a partly cloudy day predicting direct insolation becomes very difficult, particularly on short time scales (as a cloud can quickly pass in front of the sun). Reflected radiation is often assumed to come equally from everywhere on the horizontal plane with a reflectivity of 0.2 assumed for most ground or 0.8 for snow (Goswami et al., 1999). Direct insolation to a panel is scaled by the cosine of the sunlight incidence angle except that when the angle is large a significant amount of solar radiation can end up reflecting off the panel instead of reaching the cells.

For long-term solar predictions, an alternative to orbital and atmospheric models is to simply use recorded data from weather stations. For example, the National Renewable Energy Laboratory built Typical Meteorological Year (TMY) datasets for 1,020 locations in the United States and its territories (Wilcox and Marion, 2008) for use in renewable energy calculation. For any date in the constructed typical year there are hourly values for sun angle, direct and diffuse insolation, and other relevant parameters for solar power calculations such as temperature and wind. Each month for the typical year is selected to be the instance of that month in the history of the weather station that was most typical, with an emphasis on solar parameters, so a particular day cannot be said to be typical but each month can.

When predicting long-term insolation, it is important to not neglect solar panel fouling. If a robot is embedded in an environment for a long time, its panels will become coated in dust and in some parts of the world covered in snow. Dust accumulation alone can cause insolation to decrease by more than 75% if it accumulates to more than 250 grams per m^2 (Mohammad, 1993).

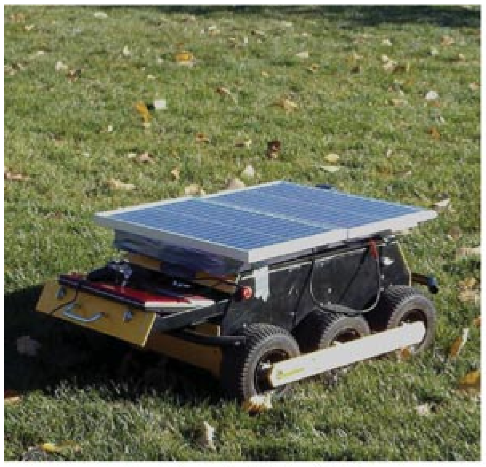
The Cool Robot team (Ray et al., 2007) used Antarctic insolation data coupled with snow reflection estimates (and estimated albedo of 90% with uniform scattering in all directions) to calculate how much sun they could expect to get on a given day from their panels. From this information they determined that vertical solar panels would be nearly optimal, and from estimates of solar panel efficiency and Maximum Point Power Tracker efficiency they calculated how much energy they could expect to get in an Antarctic day. The TEMPEST mission-level path planner (Tompkins et al., 2006) performs raytracing from the position of the sun, but it is unclear exactly how they convert their binary shadow map into estimates of solar power.

In this work we propose a data-driven approach to both the sunlight modeling problem and the power modeling problem, sidestepping most of the subtleties in short-term prediction. However, use of the TMY database for long-term power calculations is very much in the spirit of what we do in the short term, and it is in fact how we chose our solar panels (see Section 4.1 for more details on our panel selection).

3. SOLAR MODELING

In this section, we introduce the method we use to estimate how much solar power the robot will receive at a given position. As described earlier, the input for modeling the solar power is a series of solar power values associated with the corresponding robot position. Given these data, a solar map can be built through regression. We use Gaussian Process regression, which is, in general, a nonparametric regression technique. For a picture of our mobile robot and a satellite image of our testing environment, see Figure 1. For example maps that we constructed in this environment, see Figure 2.

An advantage of using regression for solar modeling is that we have an estimate for how much solar power we expect to collect at any point regardless of the amount of information we have. This estimate approaches the prior expected value as we get farther from the sampled points. Gaussian Process regression in particular outputs the entire probability distribution of a sample point, which can be valuable information for an exploration algorithm. We utilize this knowledge of uncertainty in Section 5.4.


(a) Clearpath Husky A100

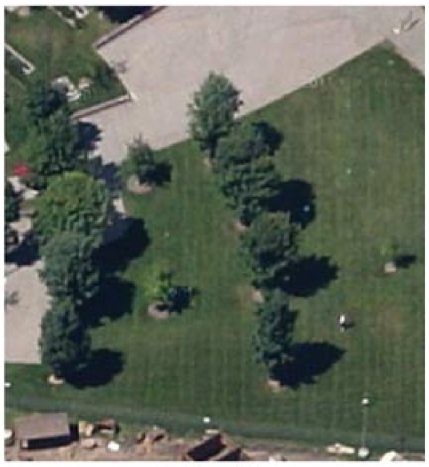
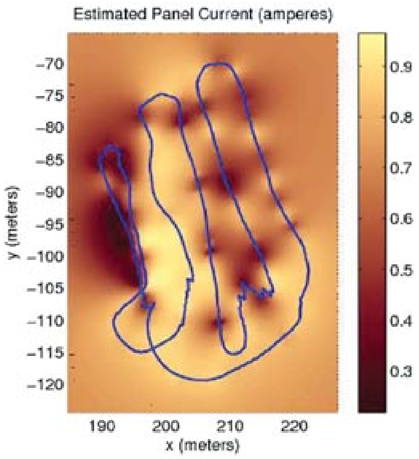
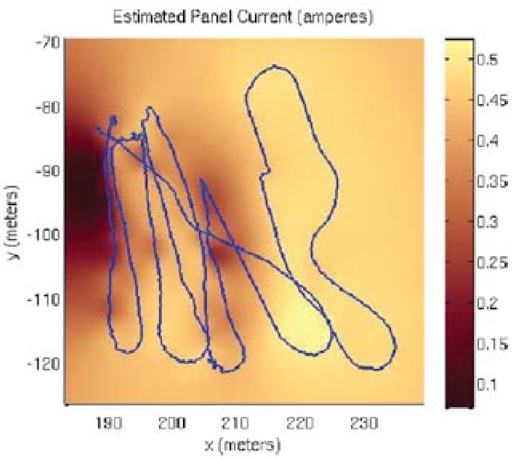

(b) Test Site

Figure 1. Our configuration of the Clearpath A100 and a top-down view of our test site near the McNamara Alumni Center.



(a) Clear Day Solar Map



(b) Overcast Day Solar Map

Figure 2. Solar map constructed for 13:42 on November 18, 2011 (a) (this was a sunny day), and solar map constructed for 11:22 on September 16, 2011 (b) (this was a cloudy day). Both solar maps are overlaid with their source paths. The cloudy map was built by sampling with only a single solar panel.

3.1. Gaussian Process Regression

A Gaussian Process (GP) is defined as a set of random variables such that any subset of the random variables has a joint Gaussian distribution (Rasmussen and Williams, 2006). GP regression is a general regression technique used to predict the most likely value of a function at any point given measured values of the function at some other points, without assuming an explicit parametric model for the function. GP regression, however, requires a suitable covariance function to model the joint Gaussian distribution for points. For more details on GP regression in general, see Rasmussen and Williams (2006).

Gaussian Processes are recently finding increasing use within the robotics community. Gaussian Process techniques have been used for localizing a mobile device using wireless signal strengths measured by the device (Ferris et al., 2007) and for the converse problem of localizing a wireless signal source using various measurements of wireless signal strength at different positions (Fink and Kumar, 2010). Gaussian Processes have been used to approximate gas concentration (Stachniss et al., 2009) and environment traversability (Martin et al., 2012). Also, the inherent estimate of uncertainty that comes from GP regression has been used to guide mapping of benthic habitats (Rigby et al., 2010) and detailed underwater structures (Hollinger et al., 2012).

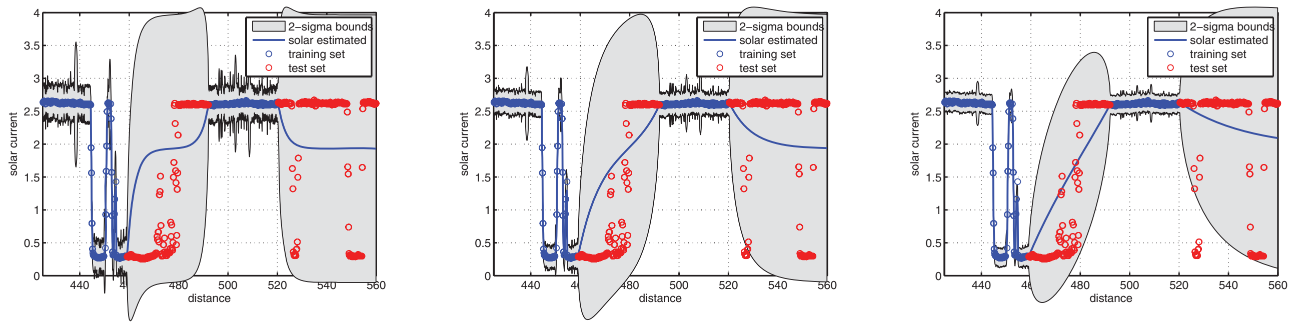
In our application, we approximate the solar field as a spatial Gaussian Process. We associate each measurement of solar power with a training position and use GP regression to predict the distribution of solar power at any desired test position. When all of the solar cells are horizontal, or if they are otherwise suitably symmetric, the rotation of the robot can be ignored in these position measurements. This makes the solar map easier to learn by eliminating a dimension along which solar power can vary. Our path planning in this paper neglects the solar map's time dependence on the changing position of the sun. This is justified when the robot stays in the same environment each day, and can therefore build a separate solar map for various discrete time segments. In Section 5.3 we examine in detail the implications of our static solar assumption.

We represent the solar field as a GP where the covariance between any two points depends only on the distance between them, through an assumed function $k(r)$. A crucial part of selecting the correct covariance function is selecting the correct length hyperparameter ℓ that scales the actual distance r. The selection of ℓ determines how quickly our estimate of solar power regresses to the prior mean, as we move away from previously sampled positions. For an illustration of this, see Figure 3.

3.2. Solar Map Construction

Our system has panels that are all aligned and horizontal, each with a maximum power point voltage of 17.2 V. These panels are connected in parallel to a 12 V battery, so our measure of panel current to the battery is directly proportional to solar intensity and also directly proportional to solar power into the system. We connect a current sensor to measure the current flowing from the panels into the battery.

The current sensor is read at every 50 ms, so the input information for our solar map is a long path with noisy 20 Hz measurements, each measurement associated with a position on the path. This accumulates to a very large number of measurements if the robot is embedded in the environment for a long time. As GP regression relies on matrix multiplication of all training points, using all measurements as individual training points becomes infeasible. Fortunately, since we only care about associating solar current to the (x, y) position, we can discard information about rotation and time and combine measurements with a similar (x, y) position. In this way, the number of measurements considered by the GP regression is bounded by the size of the environment rather than the length of time the robot is collecting data. Also, it is valuable for optimizing the hyperparameters of the GP for measured positions to be weighted equally instead of weighted in proportion to the amount of time the robot has spent at a location.



(a) length = 3 meters (b) length = 9 meters (c) length = 27 meters

Figure 3. An example of the output of 1D GP regression using $\ell = 3$, 9, and 27 m. The covariance function is the exponential function, and noise variance as well as prior mean and variance are set using the procedure in Algorithm 1.

In our implementation, we placed in a bucket all measurements that were within 0.3 m of the first measurement, and then we removed them from the list and repeated until every measurement was in a bucket. The bucket's position was set as the centroid of the positions of the measurements in it, and its value was set as the mean of the values of the measurements in it. We calculated the variance of each bucket from the variance of the measurements in the bucket, treating the bucket solar current as an average of uncorrelated random variables. Then for the regression we treated the noise variance as equal to the average of the variances of the buckets. This was again to induce more balanced weighting of different areas; if the robot had waited a long time at the same position, we did not want the bucket containing that position to be significantly more valuable than nearby buckets because still only a small portion of the possible points that could go into that bucket would have been explored. The prior mean and prior variance were computed from the mean and variance of the set of buckets. See Algorithm 1 for a pseudocode describing our bucketing procedure.

```
Input: RAWLIST ;                                        /* list of (x y solar) triplets */
TRAININGSET = $\varnothing$;
BUCKETVARIANCES = $\varnothing$;
while RAWLIST $\neq \varnothing$ do
    remove first node $s$ from RAWLIST;
    BUCKETSET = $\{s\}$;
    forall $m$ remaining in RAWLIST do
        if distance from $m$ to $s$ is $\leq d_{thresh}$ then
            add $m$ to BUCKETSET;
            remove $m$ from RAWLIST;
        end
    end
    $c$ = centroid of BUCKETSET;
    add $c$ to TRAININGSET;
    $v$ = variance of solar amplitudes in BUCKETSET;
    add $v$ to BUCKETVARIANCES;
end
Output:
TRAININGSET ;                                           /* list of (x y solar) triplets */
$\sigma_n^2$ = mean of BUCKETVARIANCES ;                /* variance of additive Gaussian noise */
$m_p$ = mean of solar amplitudes in TRAININGSET ;       /* prior mean of solar field */
$v_p$ = variance of solar amplitudes in TRAININGSET ;   /* prior variance of solar field */
```

Algorithm 1: This algorithm generates the input to Gaussian Process Regression

3.3. Covariance Function Selection

To perform GP regression, we need a covariance function. For this we considered different versions of the Matérn covariance function [detailed in Rasmussen and Williams (2006)]. The Matérn class of covariance functions is given by

$$k(r) = \frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2v}r}{\ell} \right)^{v} K_v \left(\frac{\sqrt{2v}r}{\ell} \right),$$

where v is a positive parameter that affects the smoothness of the process, ℓ is the positive length parameter, and K_v is a modified Bessel function. If v is 1/2, the function becomes the exponential covariance function, and as $v \to \infty$ the function becomes the squared exponential covariance function. Other than the exponential and the squared exponential, the most commonly used Matérn covariance functions are where $v = 3/2$ and $5/2$, so those are the covariance functions we tested in addition to the exponential and squared exponential. We considered solar data collected in the field near the McNamara Alumni Center at the University of Minnesota (see Section 4.2 for a detailed description of the test environment). This is a savanna-type environment, with scattered trees. Note that the type of shadow-casting object in the environment has little effect on sharpness of shadows—the only effect is on their position. This is because of the strong effects of partial shading (see Section 2.1). We chose this environment because we expected the shadows of its scattered trees to be difficult to learn—more difficult than the larger uniform block shadows that we would encounter near buildings or dense forest.

To optimize the Matérn function's length hyperparameter, we performed numerical gradient-descent searches to maximize the likelihood of the observed values under the assumption that the field is in fact a GP with the given covariance function. See Figure 4(a) for the optimized length hyperparameters. Then to decide which covariance function was the best, we performed cross validation on 16 solar power data sets we collected by driving manually, on various days and in various weather conditions, and looked at the resultant error. We wanted to know how well the information from a long training path would predict the solar power on a short test path, so for the cross validation we removed from the training path a continuous segment with length one-tenth that of the total path, and we observed the difference between the actual values along that segment and the values predicted by GP regression from the remaining part of the training path. This process was repeated 1,000 times for each data set. We recorded both the mean-squared error and the mean of the squared error normalized by the predicted variances at the test points. See Figure 4(b) for the squared error results and Figure 4(c) for the normalized results.

We found that using $v = 1/2$ resulted in the lowest squared error on all three cloudy day trials, and generally resulted in the lowest squared error on the sunny day trials, particularly for the trials with the most complete coverage of the domain ($v = 1/2$ was the best in 4 out of the 6 trials with more than 10,000 raw input measurements). However, the normalized error with $v = 1/2$ was usually greater than 1, occasionally by a significant margin, suggesting that the length parameter found using maximum likelihood was too high and we were somewhat overfit.

On some days the exponential covariance function resulted in the most cross-validation error by a significant margin. However on these days, as on all sunny days tested, the maps constructed with other covariance functions than the exponential tended to have problems with overshooting at the sharp boundaries between sun and shade. This sharpness was exacerbated by the strong effect of partial shading of a series of solar cells, recall Section 2.2. See Figures 4(d), 4(e), and 4(f) for examples of the overshooting problem. This overshooting made it so that the positions with the most predicted solar power were close to shadow boundaries, and therefore planned energy-minimal paths were pulled toward boundaries. As a real system always has some localization error, a better strategy is to stay away from shadows if possible. This is the behavior that results when we use $v = 1/2$ in our regression, so that is what we did when we constructed maps for path-planning purposes.

Holding $v = 1/2$, the most likely length varied between 2.05 and 12.65 m on sunny days, and between 3.68 and 18.67 m on cloudy days. This difference is because diffuse insolation dominates over direct insolation on cloudy days, and diffuse insolation varies slower than direct with changing position. See Figure 4(a) for a plot of all optimized characteristic length parameters. At first glance this seems like a large variation in length, but the data from the densest samples (trials #10 and #14-16) had optimal lengths in the tight range between 8.37 and 10.28 m. This suggests that 9 m is a good first estimate of length for a sunny day. Then if there is enough training data with the current conditions, the length should be adjusted. The only other sunny data set with more than 10,000 measurements was late enough in the day (beginning at 16:24 CDT on October 9th) that the sun was at a noticeably different position at the beginning and the end of the data set, and therefore shadows that were contiguous at any one point in time became jagged and misaligned [see Figure 5(a)]. The correct way to deal with these outdated measurements is by adding process noise, not by decreasing the length. We have fewer data for overcast conditions but we expect the roughly 16.5 m length found for the first two sets to be more typical than the 3.68 m length found for the third one.

Part of the reason for the high normalized error when performing cross validation may be that while GP regression predicts a Gaussian distribution for a given point, the true expected distribution on a sunny day is distinctly bimodal, with separate peaks for the case in which the point is in the sun and the case in which the point is in the shade. For an illustration of this, see Figure 5(b).

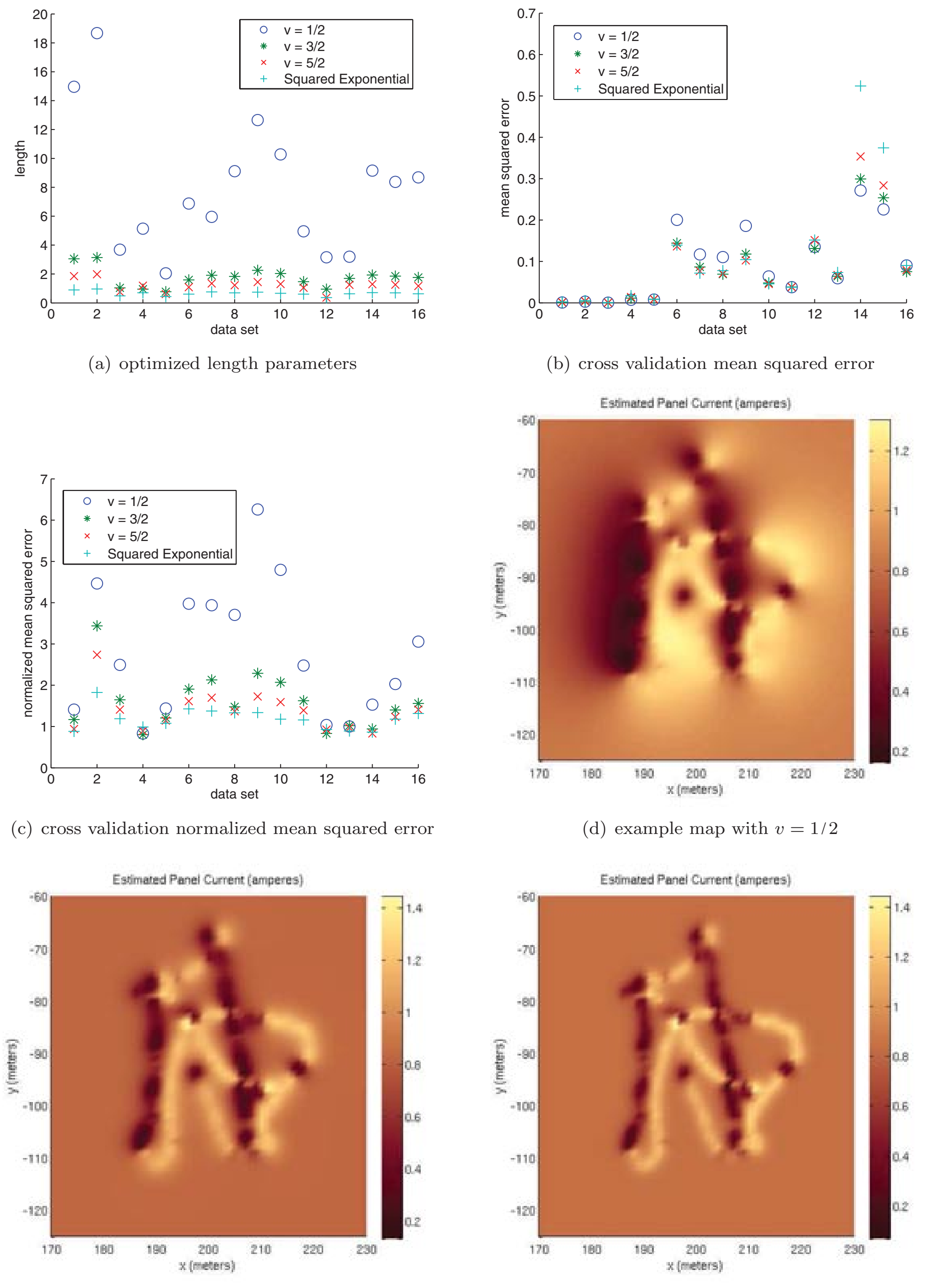
4. PATH PLANNING

In this section, we show how we use a solar map to plan the path that will reach the goal within the time limit while consuming the least amount of energy overall. It is straightforward with our approach to prune all paths that completely deplete the battery midtrip. However, as the time scales considered in this work are fairly short, this pruning was not necessary. We envision that our algorithm will be used as a subroutine for a global mission planner that attempts to maximize lifetime. Such a planner would consider battery state and weigh the cost vs. benefit of going to particular positions at particular times, and would likely consider separate solar maps for different times of the day. The purpose of this planner is to allow the cost of going to a particular position at a particular time to be estimated accurately.

4.1. Our System

The chassis of our system was a Husky A100, built by Clearpath Robotics.[1] The A100 is a six-wheel, two-motor, differential drive machine. The datasheet mass is 35 kg, the maximum payload is 40 kg, and the dimensions are 0.860 m long by 0.605 m wide by 0.350 m tall. In its experimental configuration the A100 was powered by a single lead-acid battery that was nominally 12 V and 21 A hours. See Figure 1(a) for a photo of the A100 during one of our experiments.

[1]http://www.clearpathrobotics.com/



(a) optimized length parameters

(b) cross validation mean squared error

(c) cross validation normalized mean squared error

(d) example map with $v = 1/2$

(e) example map with $v = 3/2$

(f) example map with $v = 5/2$

Figure 4. (a), (b), and (c) are the maximum-likelihood length parameters, cross-validation mean-squared error, and normalized cross-validation mean-squared error, for the same 16 data sets. The first three are cloudy days, and the rest are sunny days. The last three are the high-quality data sets used to make Figure 11, and the other trials with more than 10,000 input measurements are #5 and #10. (c), (d), and (e) are three maps built for 12:12 on November 12, which is index 6 in the cross-validation plots. Although the Matérn with $v = 5/2$ has the least cross-validation error, the exponential covariance function ($v = 1/2$) is the only covariance function tested to not suffer from significant overshooting at boundaries.

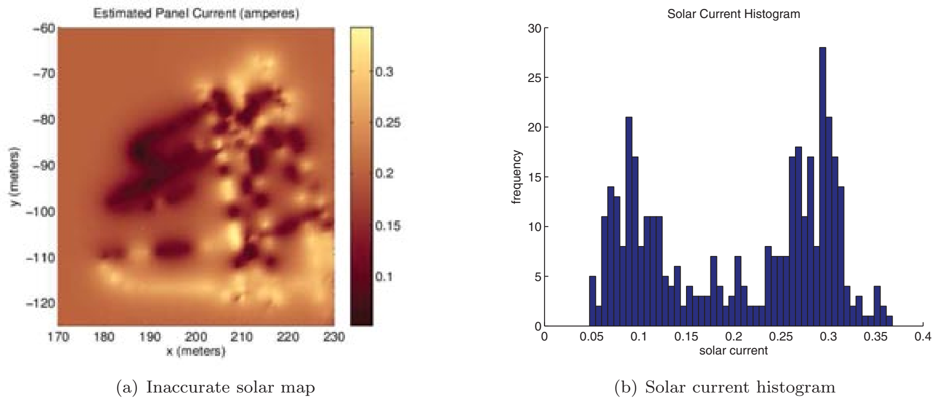


Figure 5. Solar map constructed from measurements starting at 16:24 on October 9th, 2011, and a histogram of bucket values from these same measurements. This was late enough in the day that the positions of shadows changed noticeably in the duration that the measurements were made.

The solar panels used by our system were two SPM020Ps from Solartech Power.[2] The SPM020P supplies 20 W at the optimal voltage of 17.2 V under standard test conditions of 1000 W/m^2 insolation and a temperature of 25 °C. The panel is wired as a single series string with 36 cells in it. The dimensions are 560 × 360 × 18 (mm), and each panel nominally weighs 2.5 kg.

We placed the panels horizontally on the robot for ease of mounting, for quality in overcast conditions, and to eliminate the dimension of panel rotation in the learned solar map. Both panels were connected in parallel with the battery; therefore, solar intensity, solar panel current, and solar power into the battery were all proportional. Battery voltage and motor current measurements were provided by the A100, and summed current from both panels to the battery was measured with a Hall-effect current sensor.

From the TMY3 dataset for the Minneapolis–St. Paul International Airport, we estimated that each panel should provide on average 100.4 W hours per day in June and 17.9 W hours per day in December, if it has line of sight to the sun the whole day. If the panel is in the shade but can still see most of the sky, we would expect 40.0 W hours per day in June and 8.0 W hours per day in December. This means that with both panels and given our power-to-drive model we calibrate in Section 4.4, we would expect the A100 to be able to travel 7.2 km on an average day in June using only solar power (this analysis neglects the idle current draw of the electronics).

Localization of the robot was achieved by using an Extended Kalman Filter to fuse GPS measurements with wheel-encoder propagation.

[2]http://www.solartechpower.com/

4.2. Test Environment

Our representative test environment was the field next to the McNamara Alumni Center, on the Minneapolis campus of the University of Minnesota [see Figure 1(b)]. The field is roughly 40 m by 30 m and it is relatively flat, with uniform short grass. There are not enough nearby objects to interfere with GPS localization, but there are scattered trees that provide an interesting solar map. The ground is quite flat and the grass is maintained at a short height so the power-to-drive does not significantly change depending on position and orientation.

While our calculated power-to-drive parameters and solar map parameters are likely to change in other environments, the methodology we present here to obtain those parameters remains the same. We performed our experiments on dry days when there was no snow on the ground. We expect power to drive to significantly change in wet weather or if there is accumulated snow.

4.3. Power to Drive Model

Our robot is differential-driven, so it can turn in place, and turning is a relatively energy-intensive operation. For our robot the energy consumption of a path with a certain top speed is well represented as a short initial spike during acceleration, and then a steady cost per meter traveled. Therefore, we model the planned path as time-stamped waypoints with straight line segments connecting them, each line segment traversed at a constant speed with instantaneous speed changes between line segments. We model the energy sent to the motors as the following: At any particular speed, there is a constant cost per meter traveled C_s, a constant cost per radian rotated C_r, and an initial acceleration cost C_a. When transitioning from a nonzero speed, the acceleration cost is the C_a for the new speed minus the C_a for the old speed, but with a minimum cost of 0. This makes sense if we assume that acceleration cost is proportional to kinetic energy. We can mathematically state the cost of traversing line segment l_i:

$$cost_i = C_s(speed_i)|l_i| + C_r(speed_i)|\theta_i - \theta_{i-1}| + \max(C_a(speed_i) - C_a(speed_{i-1}), 0).$$

The cost constants as functions of speed are specific to the robot and the terrain. The terrain where our experiments were conducted was flat and uniform, so in this work we do not consider changes in elevation, friction, or rolling resistance.

The total cost of a path is given by the sum over the path $\sum_{i=0}^{n-1} cost_i$ minus the expected amount of solar energy collected while traversing the path. An idle power draw constant can be subtracted from the solar power; we do not consider idle power draw because our focus is on path planning and idle power does not affect the optimal path to reach the goal in the time scales we consider.

4.4. Power to Drive Calibration

We controlled the forward movement of the A100 by directly setting the motor voltage. We found that this method was more energy-efficient than using a closed-loop PID speed controller. For a particular motor voltage and on particular terrain, the A100 travels at a particular steady-state speed and consumes a steady amount of energy per unit distance traveled, after a brief acceleration period. To characterize the steady-state cost and acceleration cost, we drove straight at a variety of commanded motor voltages from a variety of start positions across the test domain, and we fit a line to the plot of cumulative cost vs. distance for each voltage. The slope of the line determined the steady-state cost, and the intercept determined the acceleration cost. Then we performed linear regression on the steady-state costs as functions of speed and quadratic regression on the acceleration costs as functions of top speed, and we ended up with the following equations for our parameters C_s and C_a (see Figure 6):

$$C_s = (-17.6624 * speed + 139.4576) \text{ Joules per meter},$$

$$C_a = (321.0671 * speed^2 - 285.3912 * speed + 154.9553) \times \text{ Joules to accelerate.}$$

Then to characterize turning cost, we commanded a tight left turn and tight right turn and examined the steady-state energy per radian,

$$C_r = 406.5963 \text{ Joules per radian.}$$

4.5. Algorithm

The expected value for any particular point in our solar map can be determined in closed-form, however there is no convenient closed-form model for the entire map as a whole; that is, there is no general geometric model we can use to represent our environment. Therefore, some amount of discretization of the solar map is necessary for us to do planning. It is possible in this domain to plan on a set of sampled actions or path shapes (e.g., with a sampling-based planner), but since the state space is relatively small we use a complete grid. We then perform dynamic programming to compute the optimal solution for a given resolution. We discretize both space and time, and we also have a dimension in the dynamic programming table for heading and a dimension for whether the robot is moving or the robot is waiting to account for the cost to rotate and the cost for initial acceleration. These four dimensions ensure that the output path is always optimal in its resolution, according to our power-to-drive model. The trajectories generated by our algorithm move at a constant speed when they are on Manhattan edges and another faster constant speed when they are on diagonal edges, such that the time to get from a position to any adjacent position is the same.

We observe from the output of this algorithm that if the solar intensity at the start point is not equal to the solar intensity at the end point, there can be only one wait point that we denote by W. That is, the robot moves continuously before W and cannot stop to collect additional solar at any point after W. Even if the solar intensity is equal at the start and the end, this means the path with one wait point W is only one of many optimal paths. As more time is allowed, this W changes position along the path: At first there is no time to wait anywhere and the entire path is executed at max speed. Then there is time to wait but not enough to compensate for the energy loss from having to reaccelerate, so W is selected at the start point or the end point. Then finally there is enough time to wait somewhere in the middle for long enough to recoup the extra acceleration cost and possibly enough time to allow deviation from a shortest path to a place that receives more sunlight, so W is

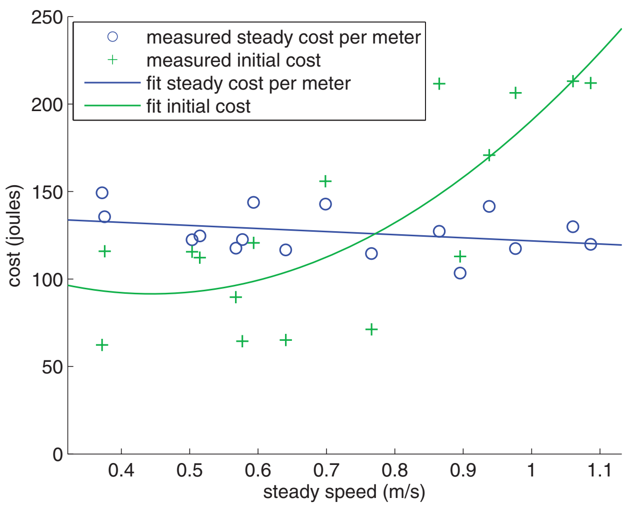


Figure 6. Power-to-drive test results.

selected at the point along the optimal path that is expected to receive the most solar radiation. Note that this single wait point optimality breaks down if we consider the possibility of overcharging or depleting the battery along the route.

4.6. Experiments

At 13:10 on February 18, 2012, we drove the A100 around the field in Figure 1(b), optimized the length hyperparameter for that dataset with an exponential covariance function, used GP regression to build a solar map, planned paths with our planner detailed in Section 4, and then executed the paths. The path used to build the solar map was a manually driven sparse coverage of the environment, which encountered each of the major shaded regions. We demonstrated in Section 3 that our method of generating the solar map is sound; here we demonstrate that our path-planning method is sound when given a reasonably accurate but low-detail solar map. The optimized length hyperparameter was 3.158 m. We later discovered that this length hyperparameter was probably too low; see Section 3.3. The A100 had some localization error even when GPS worked well, so a fairly low spatial resolution of 5 m was used. Temporal resolution was set to 8 s. To calculate the expected solar current in a grid square, the expected solar current was calculated on a higher-resolution 1 m grid and then downsampled. In addition to the planned solar-aware paths, the A100 also executed shortest paths after we removed the solar panels (slightly decreasing the power to drive due to decreased weight) from the same start position to the same end position. These paths provide a comparison, allowing us to directly demonstrate the utility of the added panels.

One of the advantages of our dynamic programming approach is that obstacles and other environmental parameters that affect the quality of a path can be incorporated into the planner simply by changing the corresponding cost in the table. In our experiments, we planned to use only the solar maps because the terrain was uniform and obstacles were sparse.

See Table I for path summaries, and Figure 7 for plots of the planned and executed paths.

On February 18, the system did not lose much accuracy by neglecting to consider the sun's movement, though the solar map was constructed for 13:10 and the last solar trial (trial E) began at 14:19. The impact of moving shadows may have been mitigated by the fact that shadows were sparse due to bare branches on the trees. In Section 5.3 we address this concern more fully.

Our power-to-drive model was reasonably accurate. It tended to underestimate power to drive but not by much: on average it missed by 396.5 J, which was on average 11.2% off from the true value. It underestimated four times and overestimated once. This indicates that our learned parameters were correct and that the A100 waypoint navigation software was not performing too many corrective turns. To get the waypoint navigation software to this state, we banned backtracking and instead considered a waypoint as reached whenever the plane perpendicular to the path was crossed. This had the effect of slightly decreasing the solar prediction accuracy, but significantly decreasing the average power to drive for a trial.

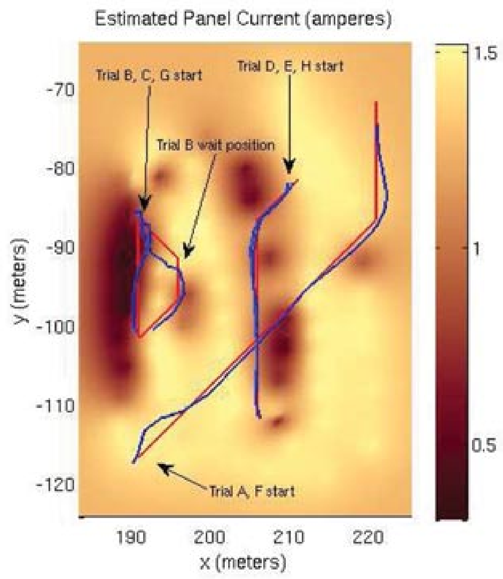
Our path planner worked well at its resolution. If we move to higher resolution, there is a danger of the following: the path planner chooses to wait in a position that has sun but due to localization error the A100 ends up waiting in the shade, and an expected good path becomes very bad. With our path planning there was very high cost to deviating from a straight path: the cost of four 45^o turns and at least 3 or 4 m increased distance. Therefore, if there is not much time the optimal path will choose to wait at the sunniest spot on the shortest path instead of deviating to a sunnier spot that is slightly off the path. This is a common problem when using a grid-based planner.

5. SIMULATIONS

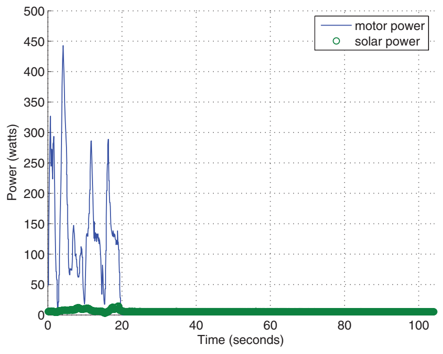
We saw from the previous section that our system performed well at a particular time on one day, but it was less useful the less duration we allowed for the path

Table I. Path execution results. On the left side of the table are the five planned and executed solar-aware paths, sorted by start position and end position. On the right side of the table are the three shortest paths executed with no panel, from the same start and end positions as the trials directly to their left. Observe that, as expected, the solar-powered robot performs best when it is allowed time to deviate from the shortest path and charge its battery in the sun.

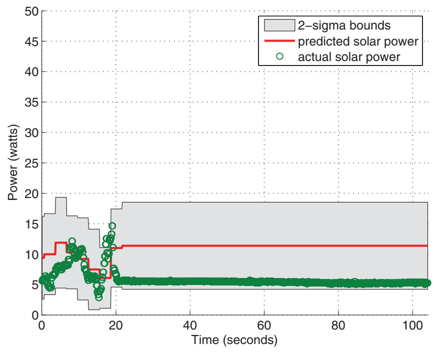
Solar Trial	Duration (s)	Expected Solar (J)	Actual Solar (J)	Expected Net Cost (J)	Actual Net Cost (J)	Control Trial	Duration (s)	Actual Net Cost (J)
A	401	7,025.5	6,974.1	577.16	744.6	F	45	6,295.4
B	400	6,606.6	6,828.6	– 3,265.9	– 3,256.7	G	19.1	2,888.1
C	104	1,148.3	611.26	879.99	2,253.4			
D	104	1,600.9	1,297.6	2,907.3	3,480.3	H	30.4	3,530.4
E	104	1,600.9	1,156.2	2,907.3	2,822.5			



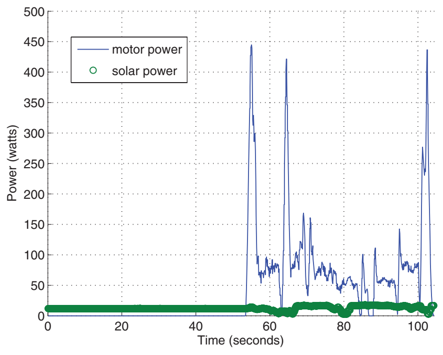
(a) Solar map at 13:10 on February 18, 2012, with planned paths in red and executed paths in blue



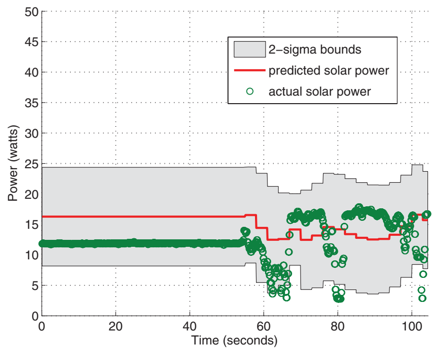
(b) Trial C Power



(c) Trial C Solar



(d) Trial D Power



(e) Trial D Solar

Figure 7. Planned solar-aware paths and example trials. Note that in trial D the planner chose to wait at the beginning given the information it had, but it turned out the position at the end of the path received more solar power.

because there was less time to stop in the sun and accumulate energy. To understand the limits of our approach, e.g., how much extra time the solar robot needed to require less energy than the nonsolar robot, we ran simulations in which we planned paths from the same starting point to the same ending point but varied the time limit. We used similar simulations to compare our planner with the naive shortest path planner.

We also wanted to understand better how the movement of the shadows during the day would affect path planning, so we ran simulations on several solar maps collected on the same day and examined the performance of paths planned using a solar map from a different time from the one on which the path is simulated.

Finally, we provide a demonstration of a data mule application in which we use our map construction and path-planning method to repeatedly find the most efficient path between two waypoints, using only the solar information available at each step.

5.1. Power Comparison

We ran simulated comparisons between our solar-powered robot using our path planner and our robot with no panels driving straight toward the destination. We built a separate power-to-drive model for the robot lacking panels so that we could compare the benefit of additional power with the cost of additional weight. We picked a start position and end position, planned the optimal solar-aware path for a range of time limits, and compared the cost to drive straight without a panel with the distribution of likely solar robot costs. For these simulations we did not consider localization errors, so we increased the resolution of our planning grid to 2 m per square, 3 s per square. For the first two simulations, we intentionally chose start and end positions in the shade to see how the system would perform under somewhat adverse conditions. The third simulation was more of a best-case analysis, and it ended in the sun.

First, we considered a robot traveling from the southwest part of the trees to the northeast part of the trees, at 13:10 on February 18 (the same day as our path execution trials). For details of this simulation, see Figures 8(a) and 8(b). The start position was (187, −106) and the end position was (207, −86). At a speed of 1 m/s we expect the baseline path to consume 3,065.0 Joules. For the solar robot to be on average more energy-efficient than the baseline, it requires at least 48 s to execute its path. This is an overall speed of 0.5893 m/s. For the solar robot to be more energy-efficient with 97.7% confidence, it requires at least 57 s, which is an overall speed of 0.4962 m/s.

Second, we considered a robot traveling south through the shade of the west line of trees, at 13:42 on November 28. For details of this simulation, see Figures 8(c) and 8(d). The start position was (193, −80) and the end position was (193, −100). At a speed of 1 m/s we expect the baseline path to consume 2,211.9 J. For the solar robot to be on average more efficient than the baseline, it requires at least 87 s, which is an overall speed of 0.2381 m/s. For the solar robot to be more efficient with 97.7% confidence, it requires at least 138 s, which is an overall speed of 0.1449 m/s.

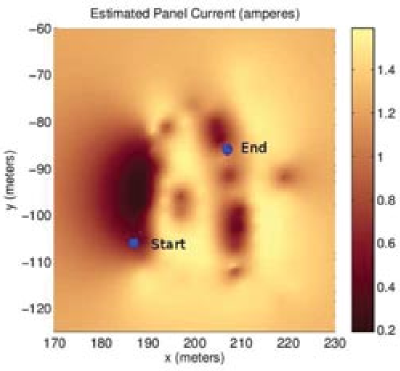
Third, we considered a robot traveling from the northwest part of the trees to east of the southeast part of the trees, at 13:41 on June 9. For details of this simulation, see Figures 8(e) and 8(f). The start position was (185, −80) and the end position was (221, −110). At a speed of 1 m/s we expect the baseline path to consume 4,947.0 J. The minimum amount of time the solar robot can take to reach the destination using our planner is 54 s (overall speed is 0.8678 m/s), and when given this time limit it requires only 4,890.8 Joules on average. For the solar robot to be more efficient than the baseline with 97.7% confidence, it requires at least 60 s, which is an overall speed of 0.7810 m/s.

The first two simulations were sunny days but they were particularly challenging for our system as sunny days go: it was the dark part of the year and the paths both started and ended in the shade. Even given the challenges, the heavy commercial solar panels were feasible additions to the Husky A100 as long as the average speed required was not greater than around 0.5 m/s. The third simulation was close to optimal for the system, and the solar system performed better than the nonsolar system even when it was allowed almost no extra time to wait in the sun.

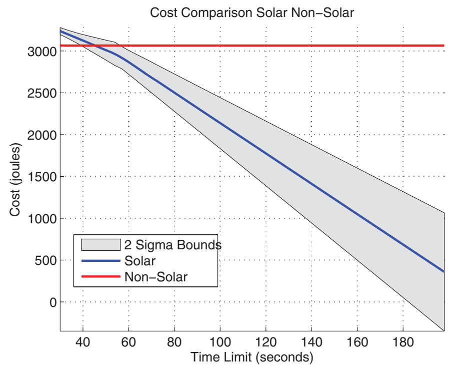
5.2. Comparison with Naive Solar

In this section, we explore the utility of using solar maps for path planning by comparing our planner with a naive planner, which plans for shortest paths but collects solar energy along the way. In general, it is difficult to say exactly how much better on average our path planner performs than a naive planner, which does not have any knowledge of the solar map; such a statement depends heavily on the underlying environment. However, we can present representative situations in which the naive strategy performs poorly compared with our planner. A shortest path planner will always perform poorly if the entire shortest path is in the shade, and depending on exactly how naive it is, it may perform poorly when both the start position and the end position are shaded, even if it passes through sun.

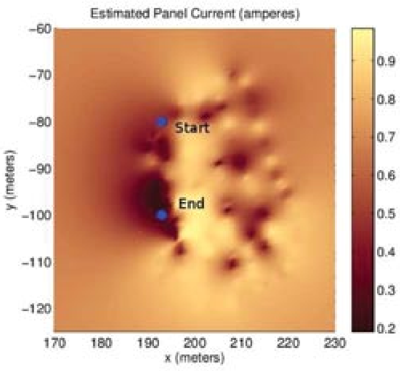
We simulated the situation in which the entire shortest path was in the shade for the solar maps built for June 9th at 16:23 and 18:20 (see Figure 11), and we found that, as expected, the more time we allowed, the better our planner performed compared with a naive strategy that performs half its waiting at the start point and half at the end point. In fact, at 18:20 the naive strategy required 297 s to even perform better than the nonsolar robot, compared with the 204 s required by our strategy. See Figure 9 for more details.



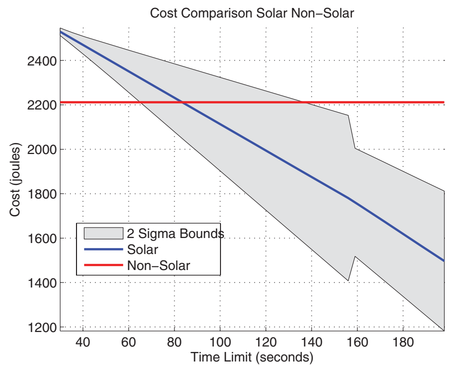
(a) Start and end positions



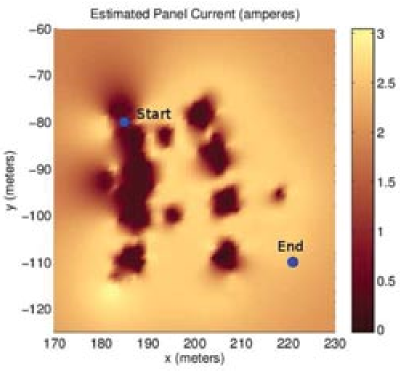
(b) Expected cost vs. time constraint



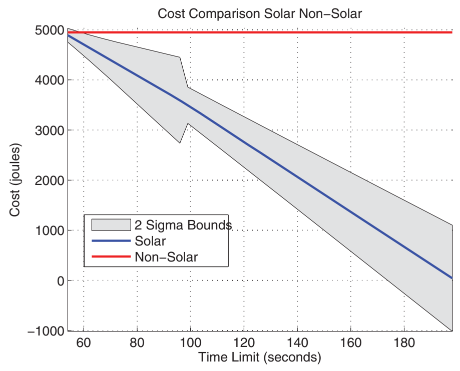
(c) Start and end positions



(d) Expected cost vs. time constraint



(e) Start and end positions



(f) Expected cost vs. time constraint

Figure 8. Simulations for 13:10 on 02-18-2012 (a and b), 13:42 on 11-28-2011 (c and d), and 13:41 on 6-9-2012 (e and f). When not much time is allowed, the weight of the solar panels ensures that the cost of carrying them is greater than the benefit of solar power. However, when the robot is allowed to wait a while in the sun, the benefit of panels can be large. The changes in slope are from changes in W, the single waiting location; as more time is allowed, a farther location can be chosen as W if it is better in terms of net energy. As the time limit increases without bound, the wait position will become the single position with the most sun.

We also performed a more rigorous test with random start and end points on the solar map constructed for June 9th at 16:23, and we found that as expected the solar-aware path is never worse than the naive path. When no wait time is allowed, the solar-aware path is only slightly better than the naive path, but when the wait time increases, the vast majority of the solar-aware paths improve upon their naive counterparts. See Figure 10 for details.

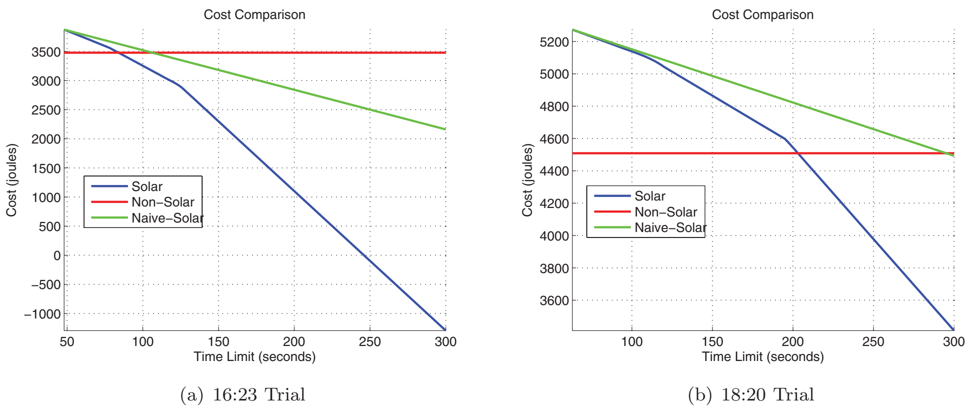


Figure 9. Comparison simulations between our planner and the naive shortest path strategy using the solar data from June 9 at 16:23 (left) and 18:20 (right). The 16:23 path was from (187, -70) to (187, -102) and the 18:20 path was from (183, 116) to (225, 116). Space discretization was 3 m and time discretization was 2 s, and the full solar data set was made available to the planner. The naive simulation and nonsolar simulation both traveled at 2/3 m/s; this was to ensure that the naive path was in the set of possible paths our planner could output.

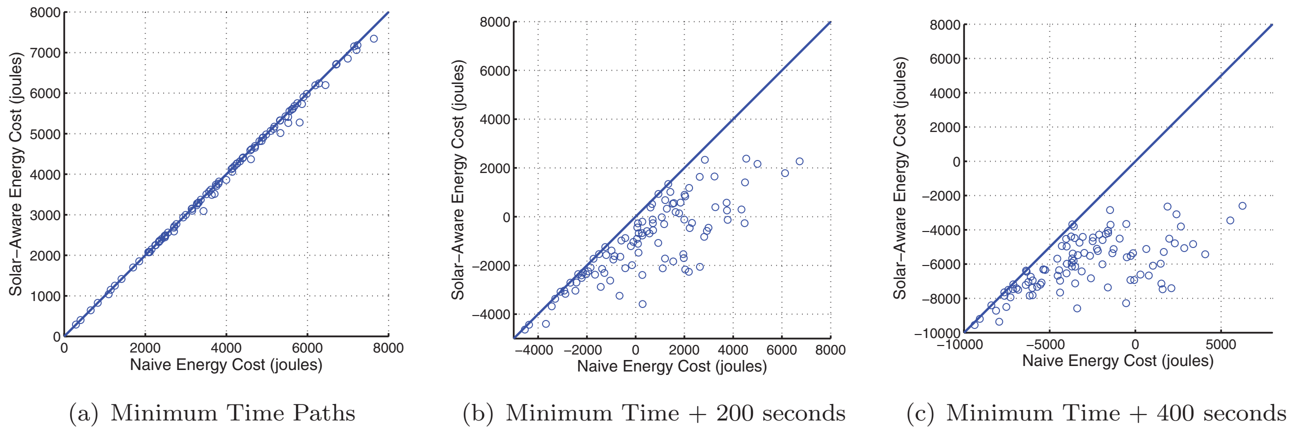


Figure 10. Results of randomized comparison simulations between our planner and the naive shortest-path strategy. The naive path was restricted to lie along the paths our planner can output; i.e., it was the shortest path on the 8-connected grid. The solar map used was June 9 at 16:23, and 100 random start and end positions were selected. Space discretization was 3 m and time discretization was 2 s. First, no extra time was allowed beyond the time required for a shortest path; then we allowed 200 s of waiting and 400 s of waiting. At our level of discretization, the naive path never performed better than the planned path, and as more time is allowed it performs increasingly worse in almost all cases. Note that even when no wait time is allowed, we can frequently perform better than naive. This is because, although we only have time for a shortest path, there are frequently several shortest paths to choose, and where naive arbitrarily picks one, our strategy makes an intelligent selection based on their relative merits.

5.3. Changing Solar Map

We have so far neglected the fact that any constructed solar map may be out of date by the time a planned path is executed on it, because the sun's position constantly changes. To examine how this affects the quality of our planned paths, we collected very-high-resolution solar data at three times on one sunny day, and we used the data to build three very-high-resolution maps. Then for each pair of maps we planned 100 paths on the map and compared the actual costs of the paths with the estimated costs. The start points, end points, and path durations were selected randomly and we used our dynamic programming path planner.

There are two main ways that our solar maps change throughout a consistently sunny day: shadows rotate around the objects that cast them, and the amount of solar radiation incident on our flat panel changes with the sine of the solar elevation angle. Accounting for the former effect is quite difficult and beyond the scope of this paper, but the latter is a relatively easy scaling adjustment. To perform this correction, we did not calculate the solar elevation angle change; this was not in the spirit of our data-driven approach. Instead we looked at the 90th-percentile solar power bucket for both maps and scaled the magnitude of the planning map to match the magnitude of the simulated map. This is justified when we can expect our robot to encounter full sun at different times of the day because in such a situation the robot should not have trouble learning the scaling factor, even if it does not know the precise solar elevation angle.

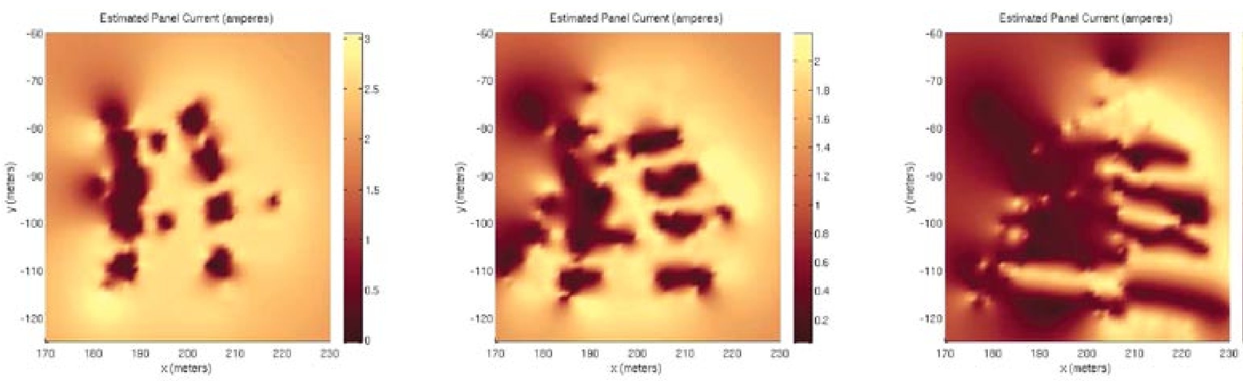
We performed simulations in which the executed path was exactly the same as the planned one on the out-of-date solar map, and there were frequent situations in which the robot passed by perfectly sunny places without stopping and then ended up waiting in the shade. This suggests that some online planning can be useful here, especially when our planning map is unreliable. So we also performed simulations in which the robot obeyed the following basic heuristic (recall that any optimal path has a single wait point that we denote here as W): If the robot encounters a point A along its path before it encounters W such that the benefit from waiting at A is greater than the expected benefit from waiting at W, the entire wait time at W is transferred to a wait at A. This heuristic ended up markedly increasing the quality of the executed paths that were planned from out-of-date solar maps.

The three solar data sets that we used for these simulations were from June 9, 2012, at 13:42, 16:23, and 18:20. See Figure 11 for solar maps constructed from these data sets. See Figures 12(a) and 12(b) for scatterplots of the expected solar energy collected during each simulated trial compared with the actual solar energy collected. We recorded the percentage of simulated trials in which the actual solar power collected was within 250 Joules of being $\geq 95\%$ of the expected solar power output by the path planner; for these results, see Table II. Note that even when we plan from and execute on the same solar map, expected solar power can differ from actual; this is because of our space discretization. Also note that it works much better to plan trajectories when the sun is low and the shadows are long and then execute them when the sun is high than vice versa. In general, our planner augmented with the waiting heuristic performed fairly well when the planning map had longer shadows than the simulated map, and relatively poorly when the reverse was true.

5.4. Data Mule Simulation

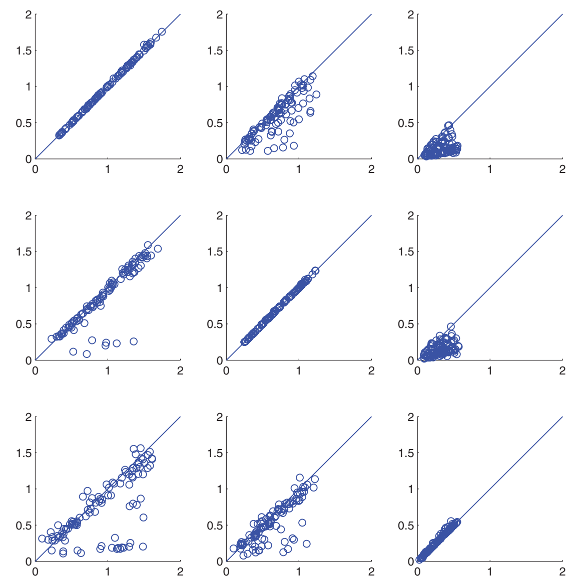
Here we present an example of how our algorithm can be useful for a practical data muling problem, where the robot starts with no solar information: Suppose we have a robot with a satellite transmitter collecting data from two sensors and transmitting it elsewhere, and this robot needs to visit a sensor once within a fixed interval. This robot is dropped at one of the sensor positions. The robot starts with no prior knowledge of the environment. However, it has perfect localization and it can measure how much solar power it is receiving at its current location. Our strategy to find a tour is the following: at each iteration, build a solar map with all the solar information, collected up to that iteration (as given in Section 3), and then plan and execute the solar-aware path using the algorithm presented in Section 4. During execution of the path, the robot will collect more solar information, which will be used to construct an improved map for the next iteration. By using the upper 2σ bound to construct the map that we feed into the planner, we can induce an exploration of uncertain terrain.

We simulated this strategy for 10 iterations (5 visits of each sensor) using the solar map constructed for July 9th at 18:20 as the ground truth, using (175, −82) and (195, −102) as the sensor positions. The robot started at (175, −82). The data collection interval was chosen to be 201 s. We used the optimized prior mean, prior covariance, and ℓ that were

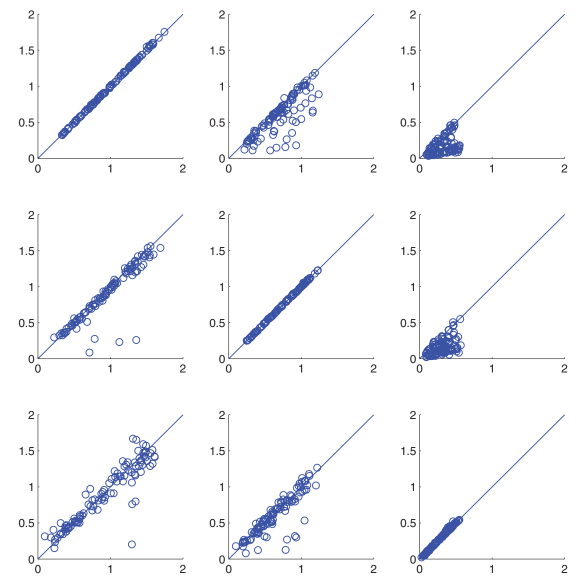


(a) 13:42 Solar Map (b) 16:23 Solar Map (c) 18:20 Solar Map

Figure 11. Three solar maps constructed for the McNamara Alumni Center on June 9, 2012.



(a) predicted vs. actual solar energy (no heuristic)



(b) predicted vs. actual solar energy (with heuristic)

Figure 12. Scatter plots of expected solar energy gain vs. actual solar energy gain for 100 simulations on the solar maps constructed for June 9, 2012. Rows are the planned maps, columns are the simulated maps, both selected from the times 13:42, 16:23, and 18:20. The x axis represents the expected amount of solar power, and the y axis represents actual simulated solar power, both divided by 10,000 Joules.

Table II. Percentage of simulated trials with $\geq$ 95% of expected solar energy. The row selects the map that was used for planning and the column selects the map on which the plan was executed. The left side plans were executed exactly and the right side plans were executed with a wait heuristic where, if a position with more sun than the expected wait position W was encountered before W, all waiting was performed at this position instead of at W.

	13:42	16:23	18:20	13:42	16:23	18:20
13:42	100%	51%	16%	100%	59%	26%
16:23	78%	100%	16%	83%	100%	24%
18:20	55%	52%	100%	77%	74%	100%

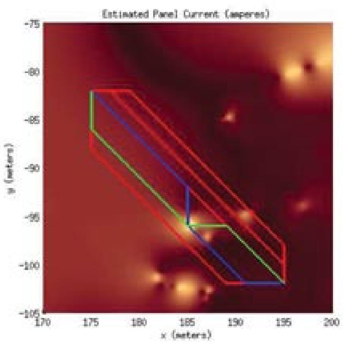
computed for that solar map. Our strategy converged at paths six and seven (marked by blue and green paths in Figure 13); after this point it did not explore anymore and instead went on path 6 whenever it was going northwest and path 7 whenever it was going southeast. Figure 13 presents further details.

This simulation shows that our strategy can be useful for this specific realistic situation.

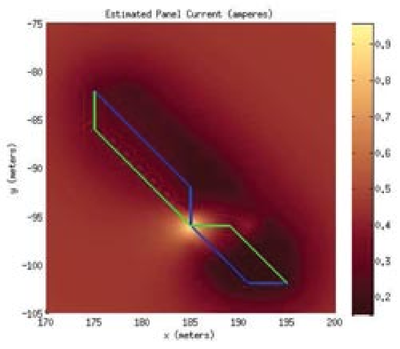
6. CONCLUDING REMARKS AND FUTURE WORK

In this work, we presented a new data-driven method to construct a map of predicted solar power. We demonstrated a system that used our method to plan energy-minimizing paths, and we examined the conditions that are necessary for our approach to be feasible. In the end, we achieved energy-minimizing, time-limited path planning that was aware of important solar factors, and we did this without the addition of any sensing hardware beyond the Hall-effect current sensor and the GPS system. We compared our approach with the approach of using a robot without a panel, and we characterized some of the situations in which we perform better and worse. Also, we specifically compared our approach with the naive strategy by which the shortest path is planned without any knowledge of the solar map, and we demonstrated the extent of our planning benefit for specific examples of start and end positions and also a variety of randomly selected start and end positions. The fundamental result is, intuitively that the more time that can be allowed to go from point A to point B, the greater the benefit of using a solar panel and the greater the benefit of planning solar-aware paths. Our methodology should provide a useful starting point for anyone building an outdoor mobile robot with solar energy harvesting capabilities.

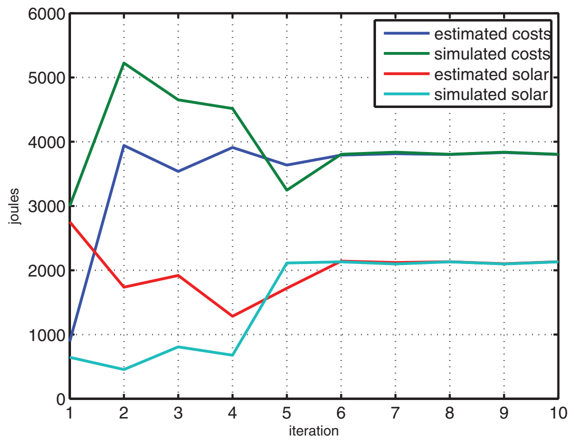
A weakness of our approach is that we do not currently have a good method to account for the fact that the solar map changes over time and instead we must construct entirely independent solar maps for different time windows. In our future work, we will investigate ways to intelligently take into account the movement of the sun when planning energy-minimizing paths.



(a) Ground truth solar map



(b) Partial solar map



(c) Chart comparing predicted and actual cost and solar current

Figure 13. The results of our data muling simulations. In (a) is the ground truth solar map used for the simulations, overlaid with the explored paths in red and the converged paths in blue and green (blue is the path from northwest to southeast, and green is the reverse return path). In (b) is the solar map estimated by the planner at the end of the 10 iterations. In (c) is the chart of predicted and actual net cost and solar current. Note that in step 5 a patch of sun is discovered that is sunnier than the 2σ bound in the map at step 4 as seen in (c). Note also that the converged cost is greater than the cost in steps 1 and 5 because step 1 began with the robot facing in the direction it needed to go and step 5 began with a more favorable angle than could be achieved later (going from the end of 4 to the beginning of 5 required only a quarter turn, but going from the end of 6 to the beginning of 5 would have required a half turn).

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant numbers 1111638, 0916209, 0936710, and 0934327.

REFERENCES

Bhadauria, D., Tekdas, O., & Isler, V. (2011). Robotic data mules for collecting data over sparse sensor fields. Journal of Field Robotics, 28(3), 388–404.

Carsten, J., Rankin, A., Ferguson, D., & Stentz, A. (2007). Global path planning on board the Mars exploration rovers. In IEEE Aerospace Conference, 2007 (pp. 1–11).

Derenick, J., Michael, N., & Kumar, V. (2011). Energy-aware coverage control with docking for robot teams. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3667–3672).

Dunbabin, M., & Marques, L. (2012). Robots for environmental monitoring: Significant advancements and applications. IEEE Robotics & Automation Magazine, 19(1), 24–39.

Ferris, B., Fox, D., & Lawrence, N. (2007). WiFi-SLAM using Gaussian process latent variable models. In International Joint Conference on Artificial Intelligence (pp. 2480–2485).

Fink, J., & Kumar, V. (2010). Online methods for radio signal mapping with mobile robots. 2010 IEEE International Conference on Robotics and Automation (pp. 1940–1945).

Goswami, D. Y., Kreith, F., & Kreider, J. F. (1999). Principles of Solar Engineering, 2nd ed. Taylor & Francis.

Hollinger, G. A., Englot, B., Hover, F., Mitra, U., & Sukhatme, G. S. (2012). Uncertainty-driven view planning for underwater inspection. 2012 IEEE International Conference on Robotics and Automation (pp. 4884–4891).

Jensen, E., Franklin, M., Lahr, S., & Gini, M. (2011). Sustainable multi-robot patrol of an open polyline. In 2011 IEEE International Conference on Robotics and Automation (pp. 4792–4797).

Kim, C., & Kim, B. (2007). Minimum-energy translational trajectory generation for differential-driven wheeled mobile robots. Journal of Intelligent & Robotic Systems, 49(4), 367–383.

Klucher, T. M. (1979). Evaluation of models to predict insolation on tilted surfaces. Solar Energy, 23.

Liu, S., & Sun, D. (2011). Optimal motion planning of a mobile robot with minimum energy consumption. In 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (pp. 43–48).

Lorenzo, E., Araujo, G., Cuevas, A., Egido, M., Minano, J., & Zilles, R. (1994). Solar Electricity: Engineering of Photovoltaic Systems, 1st English ed. Earthscan Publications Ltd.

Martin, S., Murphy, L., & Corke, P. (2012). Building large scale traversability maps using vehicle experience. In International Symposium on Experimental Robotics.

Mei, Y. (2006). Energy-efficient mobile robots. Ph.D. thesis, Purdue University.

Mohammad, S. (1993). Degredation of photovoltaic cell performance due to dust deposition on to its surface. Renewable Energy, 3(6), 585–590.

Plonski, P. A., Tokekar, P., & Isler, V. (2013). Energy-efficient path planning for solar-powered mobile robots. In Khatib, O., Kumar, V., and Sukhatme, G., editors, The 12th International Symposium on Experimental Robotics. Springer.

Rasmussen, C., & Williams, C. (2006). Gaussian Processes in Machine Learning. MIT Press.

Ray, L., Lever, J., Streeter, A., & Price, A. (2007). Design and power management of a solar-powered â cool robot for polar instrument networks. Journal of Field Robotics, 24(7), 581–599.

Rigby, P., Pizarro, O., & Williams, S. B. (2010). Toward adaptive benthic habitat mapping using. Journal of Field Robotics, 27(6), 741–758.

Sauze, C., & Neal, M. (2011). Long term power management in sailing robots. In OCEANS, 2011 IEEE - Spain (pp. 1–8).

Shillcutt, K. (2000). Solar based navigation for robotic explorers. Ph.D. thesis, Robotics Institute of Carnegie Mellon University.

Stachniss, C., Plagemann, C., & Lilienthal, A. J. (2009). Learning gas distribution models using sparse Gaussian process mixtures. Autonomous Robots, 26(2-3), 187–202.

Sugihara, R., & Gupta, R. (2009). Optimizing energy-latency trade-off in sensor networks with controlled mobility. In INFOCOM 2009, IEEE (pp. 2566–2570).

Sun, Z., & Reif, J. (2005). On finding energy-minimizing paths on terrains. IEEE Transactions on Robotics, 21(1), 102–114.

Temps, R. C., & Coulson, K. L. (1977). Solar radiation incident upon slopes of different orientations. Solar Energy, 19, 179–184.

Threlkeld, J., & Jordan, R. (1958). Direct radiation available on clear days. ASHRAE Trans., 64, 45.

Tokekar, P., Karnad, N., & Isler, V. (2011). Energy-optimal velocity profiles for car-like robots. In 2011 IEEE International Conference on Robotics and Automation (pp. 1457–1462).

Tompkins, P., Stentz, A., & Wettergreen, D. (2006). Mission-level path planning and re-planning for rover exploration. Robotics and Autonomous Systems, 54(2), 174–183.

Wang, G., Irwin, M., Fu, H., Berman, P., Zhang, W., & Porta, T. (2011). Optimizing sensor movement planning for energy efficiency. ACM Transactions on Sensor Networks (TOSN), 7(4), 33.

Wilcox, S., & Marion, W. (2008). Users Manual for TMY Data Sets. Technical report.