Autonomous Robots and Environmental Mapping

M. Horning, M. Lin, S. Srinivasan, S. Zou

August 14, 2014

Abstract

Compressed sensing is a technique used to reconstruct sparse signals and images while sampling well below the Nyquist rate. Commonly used in fMRIs and medical imaging, we aim to use this technique to allow autonomous robots to map piecewise constant areas of interest within a larger environment. In our experiment, we programmed a robot equipped with a reflectance sensor to travel along straight-line paths on a black testbed with white regions of interest. The robot integrated sensor readings and sent that sum to a remote server after each path. Each data point consists of the robot's start and end positions, tracked by an overhead camera, and the integral found along that path. Once all the data has been collected, the environment is reconstructed by minimizing the sum of the data fitting term and an L1 penalty term. When using an adaptive path selection scheme, preliminary simulations of our experiment show reconstruction of 100x100 images with only 100 data points (relative err. = 0.28), and we have begun trials with data collected by the robot.

Contents

1 Introduction

Environment mapping is amongst the foundational problems of mobile robotics. In order for an autonomous vehicle to operate effectively within an environment,

The task of mapping an unknown environment is well-studied in the area of autonomous robotics.

Compressed sensing describes a technique in which a signal is reconstructed by solving an underdetermined linear system. Doing so allows a signal to be found from a relatively small amount of data.

2 Lab Setup

2.1 Testbed

The setup in the UCLA Applied Mathematics Laboratory (AML) used three main components: an overhead camera with a tracking system, a PC computer, and the robot vehicle. The testbed is a 1.5 m x 2.0 m rectangular area made of black asphalt felt paper. The cars' positions are tracked by 2 overhead Imaging Source DMK 21F04 1/4 Monochrome CCD cameras with a resolution of 640 x 480 pixels. They have a frame rate of 30 fps and are connected to an image processing PC via firewire cable. The cars are identified using an OpenCV contour searching function that recognizes black and white ID tags, or "hats", that are fixed on top of the vehicle, giving the robot's current position and orientation [**?**]

2.2 Robot

2.3 Video Camera

2.4 Server

2.5 Putting it all together

3 Models and assumptions

Our model and lab setup were designed with certain constraints in mind. The model is designed for robots with limited on board storage and computing power. Another limitation is bandwith and infrequent communication with the main server

Without loss of generality, we assume the area to be explored is a rectangle denoted by Ω, and the interested environment variable $u(x, y)$ is a piece-wise constant function defined on Ω. See Figure 1 for an illustration. Assume the robot travels through n different paths, which are denoted by $C_1, \ldots, C_n$. Along each path C_k, the integral of environment variable u is obtained, which is denoted by g_k. The path integrals are written as

$$g_k = \int_{C_k} u(x, y)\, \mathrm{d}\Gamma,\ k = 1, \ldots, n. \tag{1}$$

For computational purpose, the whole domain is discretized into rectangular pixels (See Figure 1). Each path C_k defines a weight $a_{k,ij}$ for each pixel (i, j). If C_k does not intersect with pixel (i, j), then $a_{k,ij} = 0$, otherwise $a_{k,ij}$ is defined to be proportional to the length of the part of C_k

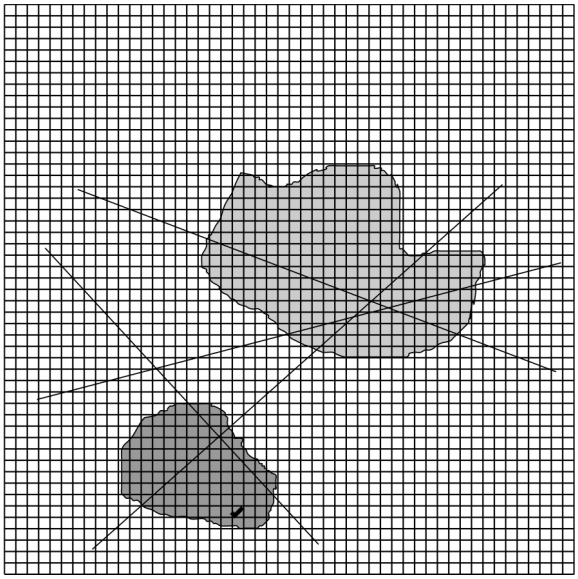


Figure 1: An illustration of the paths of robots. The shaded regions denote the area of interest (unknown), where the values of environment variable u is significantly different from the surroundings. The size of the unit pixel (each small rectangle) is determined by the accuracy of the positioning system.

that falls within pixel (i, j). It is also assumed that the value of u is a constant within each pixel (i, j), which is denoted by u_{ij}. After discretization, Eq (1) becomes

$$g_k = \sum_i \sum_j a_{k,ij} u_{ij}. \tag{2}$$

Let $g = (g_1, \ldots, g_n)^t$, $u = (u_{ij})$, then g and u have a linear relation, which is written as

$$g = Au, \tag{3}$$

where the linear operator A is specified in Eq (2). Eq (3) is the model equation, which poses an inverse problem. In this equation, g can be obtained from the experiment, A is determined by user-specified paths for the robot, which can be calculated offline. u is the variable that we want to solve. Of course, if the paths form a complete raster scan over the whole domain, then theoretically Eq (3) has a unique solution. For economical reasons, it is desirable to use much fewer paths and still being able to reconstruct a solution for u. Inspired by compresed sensing based image reconstruction techniques, it is hopeful that reconstruction for our under-determined system can be done.

4 Algorithm for solving the inverse problem

4.1 Experimental Results

5 Comparison with Other Approaches

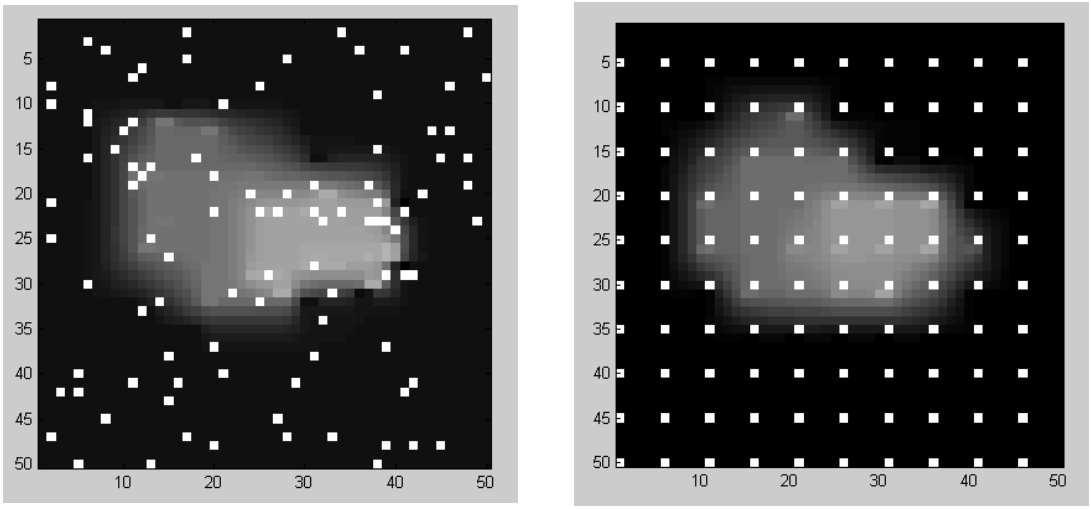
In order to judge the effectiveness of our approach, we examine other methods of collecting data and reconstructing an image. Perhaps the most intuitive manner in which to gather data about the image is to sample individual points rather than travelling and integrating along a path. As when collecting data in the form of path integrals, we would want the number of sampled points to be as low as possible. Having sampled points for data, there are then multiple approaches to recovering the image. One approach is to simply use the same Split Bregman iteration that we are using for reconstruction from path integrals, but with the individually sampled pixels. Alternatively, we can take the value found at each pixel and assign the same value to nearby pixels, a method that we refer to as pixel expansion.

Figures 2 and 3 present reconstructions of the previously shown 50x50 test image using these approaches, both with randomly sampled points and points sampled on a grid. In all cases, 100 pixels are sampled, and the sampled points are colored white.

In Figure 2, Split Bregman iteration was used with the collected data for reconstruction. The algorithm itself is unchanged, but the equation corresponding to a data point will simply give the value of a pixel rather than weighting pixels along the path. This method provides a moderately good reconstruction, especially in determining the general shape of the region of interest in the environment, but the demarcation between the shades of gray in the image is difficult to distinguish, making the image seem very blurry. This is not surprising because by sampling pixels, the only information gathered is about those specific pixels, so the space in between can gradate as quickly or slowly as the reconstruction algorithm deems fit. By comparing Figures 2a and 2b, it seems that the reconstruction is more accurate when the points are sampled on a grid. This is reasonable because by sampling on a grid, the spread of sampled points is very even and well distributed. When the points are sampled randomly, certain areas will be less thoroughly investigated than others, hindering accurate reconstruction.

Figure 3 illustrates pixel expansion. This is done by looping through every pixel in the image, and for each pixel, finding and assigning the value from the nearest pixel that was sampled. This process creates expanded pixels of a single color around each sampled pixel. When the pixels are sampled on a grid, the even distribution causes the expanded pixels to simply be squares, essentially creating a low resolution version of the image, as in Figure 3b. Figure 3a is a result of pixel expansion with randomly sampled pixels. The unpredictable spread of sampled pixels causes the expanded pixels to vary widely in shape and size, and the reconstruction does a poor job of determining the shape of the region of interest.

To compare these methods, Figure 4 plots the error of these approaches, as well as the path integral reconstruction that we are actually carrying out, with respect to the number of data points collected. The blue line represents reconstruction from randomly generated path integrals, and as desired, for a low number of data points, it has lower error than the other methods. As the number of samples grows, all methods begin to have similar error; this is expected because any method will work reasonably well with enough data. Worth noting is that while pixel expansion from samples on a grid appears to do better than random path integrals for some of the low data situations, the effectiveness of grid point expansion is highly variable, and as such, its overall usefulness is



(a) Reconstruction using points sampled randomly. (b) Reconstruction using points sampled on a grid.

Figure 2: Values are found at sampled points, and Split Bregman iteration is used to reconstruct the image.

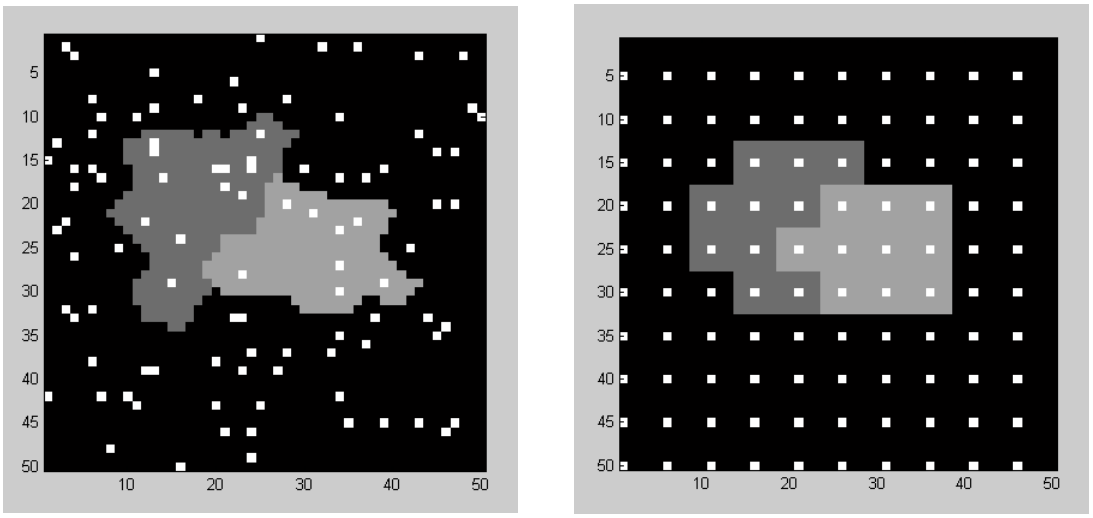
limited. The accuracy of reconstruction is very dependent on how well the grid aligns with the given image, and choosing the best grid size for a given environment would require prior knowledge of the environment. Another flaw with pixel expansion is the dependence on accurate sampling; pixel expansion will be very heavily affected by noise. If a pixel is sampled inaccurately, the entire expanded pixel associated with that sample will be inaccurately reconstructed. Using Split Bregman iteration mitigates this issue, and using path integrals further diminishes the impact that a single inaccurate sample can have.

6 Adaptive Pathing

6.1 Simulated Results

6.2 Experimental Results

7 Conclusions and Further Work



(a) Expansion of points sampled randomly.

(b) Expansion of points sampled on a grid.

Figure 3: Values are found at sampled points, and those values fill the area surrounding the sampled points.

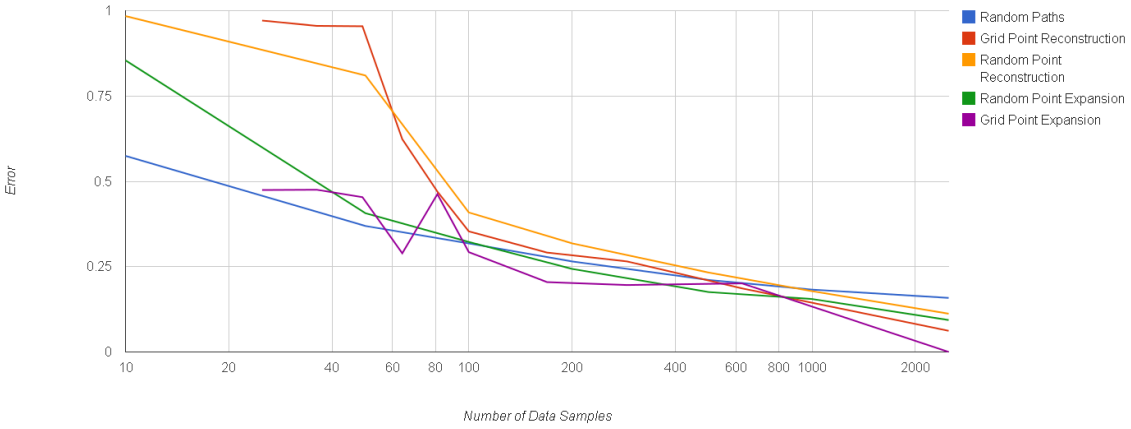


Figure 4: Plot of errors using various reconstruction methods with respect to the number of data samples used.