INTELLIGENT PLANNING AND ASSIMILATION OF AUV-OBTAINED MEASUREMENTS WITHIN A ROMS-BASED OCEAN MODELING SYSTEM

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Benjamin Davini

December 2010

COMMITTEE MEMBERSHIP

TITLE:	Intelligent Planning and Assimilation of AUV-obtained Measurements within a ROMS-based Ocean Modeling System
AUTHOR:	Benjamin Davini
DATE SUBMITTED:	December 2010
COMMITTEE CHAIR:	Chris Clark, Ph.D.
COMMITTEE MEMBER:	Paul Choboter, Ph.D.
COMMITTEE MEMBER:	Franz Kurfess, Ph.D.

Abstract

Intelligent Planning and Assimilation of AUV-obtained Measurements within a ROMS-based Ocean Modeling System

Benjamin Davini

Efforts to learn more about the oceans that surround us have increased dramatically as the technological ability to do so grows. Autonomous Underwater Vehicles (AUVs) are one such technological advance. They allow for rapid deployment and can gather data quickly in places and ways that traditional measurement systems (bouys, profilers, etc.) cannot. A ROMS-based data assimilation method was developed that intelligently plans for and integrates AUV measurements with the goal of minimizing model standard deviation. An algorithm developed for this system is first described that optimizes paths for AUVs that seeks to improve the model by gathering data in high-interest locations. This algorithm and its effect on the ocean model are tested by comparing the results of missions made with the algorithm and missions created by hand. The results of the experiments demonstrate that the system is successful in improving the ROMS ocean model. Also shown are results comparing optimized missions and unoptimized missions.

Acknowledgements

The work done in this thesis wouldn't have been possible without the inspiration, knowledge, and help of so many people.

Chris Clark (Thesis Advisor) For providing a topic and direction, and piquing my interest in the field of robotics. And also for trusting me with a $50K+ robot.

Paul Choboter (Committee Member) For all your work with ROMS and patience in (re-)explaining what I was suppossed to be doing.

Franz Kurfess (Committee Member) For providing me further experience and interest with your Artificial Intelligence classes.

Scott Lydon, Robbie Plankenhorn, and Matt Maxon For getting me hands-on experience with the Iver2 before I inherited it, as well as installing most of the hardware that I needed on the Iver2 (temperature sensor, serial connections, etc.).

Jason Felton and Tom Moylan For all of the help out on the pier. I certainly wouldn't have been able to carry out my experiments without your assistance.

Mom and Dad For the always-positive 24+ years of support you have given me. I finally finished.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The area of robotics is a large field with many areas of research and even more applications. Motion planning, mechatronics, localization, mapping, and artificial intelligence are just a few of the many fields under the umbrella of robotics. Robots are used in a variety of settings, from small domestic settings (floor vacuums), to applications in the sky (i.e., unmanned aerial vehicles) and underwater (i.e., autonomous underwater vehicles).

Autonomous Underwater Vehicles (AUVs) are natural successors to the manned underwater exploration devices and tethered underwater robots still in common use today. They offer a number of obvious advantages; most importantly, they reduce the need for manned underwater exploration, which can be a dangerous feat considering temperature and pressure at depth. Additionally, they can explore tighter spaces, move faster, and navigate autonomously. They are not strictly better, of course - disadvantages include a limited underwater lifespan due to the fact that the robot has to provide its own power and the higher likelihood of losing the robot due to errant motion planning or other malfunctions. However, if correctly configured and built, the advantages of such a system become great.

One application of an AUV is in the rapid acquisition of ocean data - depth, temperature, salinity, current speeds, etc. This data can be introduced into an ocean model to reduce the error in these ocean models. Acquired data can also be used for reports on global warming, marine habitats and global ecosystems.

This thesis seeks to explore the use of an AUV to most efficiently gather optimal data in order to reduce the standard deviation, or uncertainty, in an ocean model of part of the California coast line. A system is described that can facilitate AUV data assimilation. An algorithm is developed that does a rapid off-line search of the model to create a mission path that covers the most points of highest temperature standard deviation. This mission is executed by an AUV that collects temperature data along the mission path. The reduction in temperature standard deviation resulting from this data is compared to that gathered from other missions that were developed without using the path planning algorithm.

Chapter 2 is devoted to background information related to AUVs and an ocean modeling system called ROMS. Chapter 3 is an analysis of related works. Chapter 4 identifies the problem statement and offers a mathematical representation of the problem. Chapter 5 proposes a solution to the problem. Chapter 6 discusses the specific components used to solve the problem. Chapter 7 details the optimized and unoptimized Iver2 AUV missions. Chapter 8 presents the data acquired from each mission. Chapter 9 seeks to explain the data and how it relates to the problem statement. Finally, Chapter 10 presents a conclusion and discussion of future work.

Chapter 2

Background

There are a number of different technologies and areas of research that are relevant and symbiotic to the goal of this thesis. In this chapter, background information on these technologies and fields of research is provided.

2.1 Regional Ocean Modeling System

The Regional Ocean Modeling System, or ROMS, is a mathematical model capable of "simulating currents, ecosystems, biogeochemical cycles, and sediment movement in various coastal regions" [11]. It is an instance of a larger class of atmospheric and geophysical models, and one of many Oceanic General Circulation Models (OCGMs) [22]. ROMS was primarily developed by researchers at Rutgers University and UCLA in the late 1990s (www.myroms.org).

ROMS can model areas as large as the Earth's oceans, or as small as a bay (resolution from a few meters to hundreds of kilometers; scale from 10,000 km down to a few hundred meters). Typically, the desired output of the models - the

estimate of the ocean region - is a collection of the region's state variables. These variables are estimates of temperature, salinity, sea surface height, and velocity. Besides an estimate of each of these variables, ROMS can also provide a measure of uncertainty about these variables in the form of *standard deviation*.

ROMS utilizes a number of complex mathematical equations (the partial differential equations of fluid dynamics) in combination with acquired data (via satellite, stationary observatories, AUVs, etc.) to provide an estimate of ocean parameters within the ocean region being modeled. While the partial differential equations used in ROMS are beyond the scope of this thesis, a high-level overview of ROMS will be given in 2.2.1. ROMS relies upon data assimilation, which is explained below in Section 2.2.

2.2 Data assimilation

This thesis was motivated by the desire to provide a more accurate model of ocean parameters (e.g., temperature, salinity, ocean current) in the bay near Avila Beach, California. ROMS (described above), the system used to develop this model, relies upon the equations of fluid dynamics to propagate the states of the system forward and provide forecasts of the ocean in the future. It also relies upon real, observational data, to provide initial conditions and corrections to this model. The observational data is introduced and integrated into ROMS through a process called data assimilation.

According to Ding Wang,

> Ocean Data Assimilation refers to the quantitative estimation of marine fields of interest by melding data and dynamics in accord with their specific uncertainties. [29]

Data assimilation serves as a method to estimate the variables of an environment in question (e.g., salinity in sea water, or humidity in air, etc.) by combining observational data with the equations or principles that govern the medium being studied. Data assimilation is used primarily in the field of geoscience - in weather forecasting, atmospheric data analysis, or oceanographic data analysis. However, nearly any geophysical fluid system governed by a system of equations can benefit from data assimilation [28].

At a high level, data assimilation is an iterative process that in each iteration takes observational data and combines it with a forecast (the current estimated state of the model; e.g., one provided by ROMS). The combination of model data and actual observational data produces an *analysis*, which provides the best guess of the current system. The analysis is then fed back into the model, which will then produce another forecast, against which additional observational data will be compared. The ROMS specific use and implemenation of data assimilation is described and illustrated in detail below.

2.2.1 Data assimilation in ROMS

Given a set of initial conditions, boundary conditions, and surface wind (a forcing function), ROMS can give an estimate of the ocean system at some later time T. It does this by moving the governing primitive equations forward in time.

Any real ocean data, or observations, acquired in the region between $t = 0$ and $t = T$ will likely conflict with the model, no matter how "good" the model. In an attempt to reconcile these differences, ROMS corrects itself by reducing a cost function. This cost function is incremental; its solution (minimum) is discovered

iteratively. Given a model

$$\mathbf{z} = [T_0\ S_0\ u_0\ v_0\ \cdots\ T_n\ S_n\ u_n\ v_n] \tag{2.1}$$

this cost function J is represented by the following equation:

$$J = \frac{1}{2}\delta\mathbf{z}^T\mathbf{D}^{-1}\delta\mathbf{z} + \frac{1}{2}(\mathbf{G}\delta\mathbf{z} - \mathbf{d})^T\mathbf{R}^{-1}(\mathbf{G}\delta\mathbf{z} - \mathbf{d}) \tag{2.2}$$

where $\delta\mathbf{z}$ is the control vector (initial condition increments), $\mathbf{D}$ is the background error covariance matrix, $\mathbf{R}$ is the observation error covariance matrix, $\mathbf{G}$ is the tangent linear model sampled at the observation points, and $\mathbf{d} = (\mathbf{y} - H(\mathbf{x}^b(t)))$ is the innovation vector that represents the difference in observations $\mathbf{y}$ and the model at the times and locations of the observations [20].

J can be minimized by solving:

$$\partial J/\partial\delta\mathbf{z} = \mathbf{D}^{-1}\delta\mathbf{z} + \mathbf{G}^T\mathbf{R}^{-1}(\mathbf{G}\delta\mathbf{z} - \mathbf{d}) \tag{2.3}$$

ROMS essentially makes "compromises" between the model and the observations in order to reach an equilibrium. J is the metric through which this equilibrium is reached. The process can be illustrated as seen in Figure 2.1.

In the first step (Non-linear Forward Model), state variables $\mathbf{z}$ (e.g., temperature) are propagated forward in time to T by numerically solving the ROMS primitive equations. The data assimilation process is then started; this process is repeated as many times as possible or until convergence, where J is minimized. In the data assimilation process, a linearized version of the forward model, called the tangent linear model, is run forward to T. This model is sampled at the same

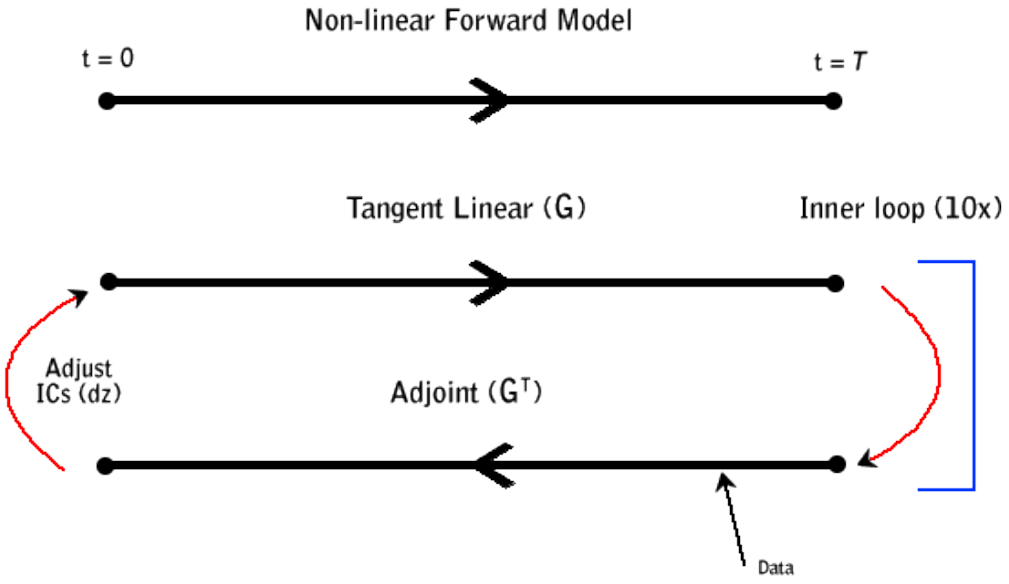

Figure 2.1: ROMS data assimilation process

time and place as the actual observations. It is then run backwards in the adjoint model to see what initial conditions would have resulted in the $\mathbf{z}$ at the end of time T, as determined by iteration. The observations are compared to the sample points at the appropriate time in the model. A method similar to least squares is used to fit the model as it moves backwards. The adjoint model will arrive at $t = 0$ with slightly different initial conditions than the previous tangent linear model. These initial conditions are integrated into another tangent linear run; the whole process (tangent linear + adjoint) is then repeated until convergence.

2.3 Autonomous underwater vehicles

Autonomous underwater vehicles, or AUVs, are part of a larger group of unmanned underwater vehicles, whose applications extend from commercial (e.g., oil and gas surveying), to military (e.g., mine countermeasures or anti-submarine warfare) and scientific (e.g., oceanographic surveying) [7].

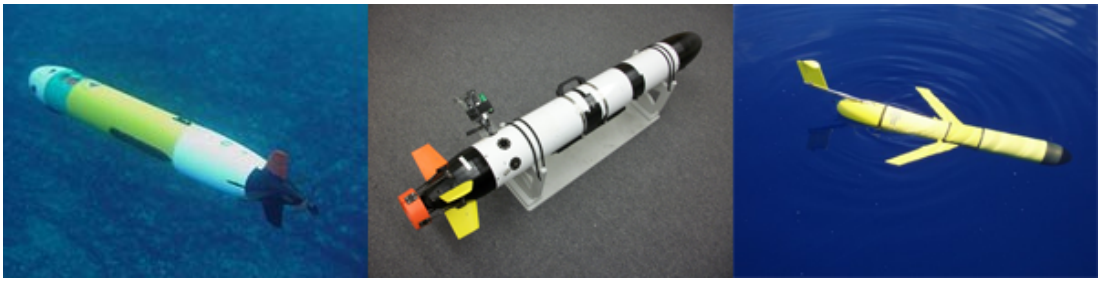

Figure 2.2: The REMUS, Iver2, and Slocum Glider AUVs

These underwater vehicles have come into existence in only the last 50 years or so, and the designs of these vehicles are numerous. AUVs typically come equipped with a number of sensors, communication equipment, navigational abilities, a power system, and a propulsion system. For minimal drag underwater, AUVs are often designed in the shape of a rounded cylinder, like a torpedo, or biomimetic (life-like) so that they may blend in with other animals in their surroundings [6]. Examples of different types of AUVs can be seen in Figure 2.2.

Examples of scientific payloads (sensors) that may be found attached to an underwater vehicle include but are not limited to Acoustic Doppler Current Profilers (ADCPs) - a sonar used to measure current velocities [30] - temperature and salinity sensors, and oxygen sensors [8]. Acoustic modems, global positioning systems, digital compasses, 802.11 wireless transceivers, sonar transceivers, and satellite phones are common technologies or instruments used to communicate with and localize the AUV [5, 4]. The AUVs are often powered by rechargeable

batteries (lithium ion, alkaline, silver-zinc, etc. [3]), though non-rechargeable batteries can be found in use for extended missions.

Underwater vehicles are primarily propelled in one of two ways. The most common method is through one or more propellers located at or near the end of the vehicle's chassis. The propeller may be uncovered, as in most boats, or it may be surrounded by a sheath like the Iver2 as in Figure 6.1 below. The propeller is used for forward and backward motion; an AUV will often include fins to control its roll, pitch, and yaw. Another method of propulsion, utilized by some AUVs (e.g., [24]), is "gliding". An underwater glider uses small changes in its buoyancy in combination with its wings to convert vertical motion (floating and sinking in the water) to horizontal movement. The key advantage of this type of propulsion is the endurance provided (months of travel), which comes at the cost of speed (.2-.4 m/s) [14]. (The Iver2 AUV used in the experiments for this thesis travels at 1.3 m/s.)

Chapter 3

Related work

This chapter discusses work and research in the area of marine data collection, including the use of stationary observatories as well as AUVs. It also discusses AUV path planning.

3.1 Marine data collection

The first global marine research exercise was undertaken by the British warship HMS Challenger only 125 years ago; the expedition gathered temperature data, water chemistry, bathymetry measurements, and other data from across the world in its four year journey [10, 27]. Since then, the field of oceanography has enjoyed a flux of interest - educational institutions, governments and militaries, and corporations have invested time and money into being able to better understand the ocean. The reasons for this interest are many-fold - commercial, military, or just curiosity - but the overall goal is the same: a better understanding of the ocean.

As in any other empirical field, observations are required in order to better understand the subject being studied. Oceanography, unlike many other sciences, faces the problem of the sheer enormity of the space being studied. Though there is continuous observation of the seas by satellite as well as thousands of free-drifting profiling floats (buoys with scientific instrumentation), the ocean remains under-sampled, including coastal regions where the water is shallow and ocean movement faster [29, 18]. For this reason, it is important to take measurements in places and/or ways that will offer the best information. *Best information*, of course, is subjective to the goal that one is trying to accomplish.

For this thesis, the *best information* are the ocean temperature measurements that reduce the variability in the ocean model discussed in Section 6.1.7. This section discusses related work in the area of marine data collection.

3.1.1 Optimizing fixed observational assets in a coastal observatory

In [16], Frolov et al. discuss the optimal placement of fixed (stationary) observatories in the Columbia River estuary and plume. Utilizing a statistical method called the Best Linear Unbiased Estimator (BLUE), they propose an optimal experiment design method, and then apply this method to design an optimal observational array for the area. They verified their proposed array designs by answering questions regarding theoretical limitations and the model of the Columbia River estuary and plume. In their conclusion, they showed that they could improve data assimilation accuracy with the appropriate placement of a single salinity sensor.

3.1.2 Autonomous Ocean Sampling Network

In Summer 2003, researchers met in Monterey Bay, CA with the obervational goal of developing "a capability to coordinate a diverse collection of manned and unmanned observing platforms within the context of data-assimilating models to form a powerful and efficient observing system..." [13]. Dozens of AUVs, research vessels, satellites, and moorings collected temperature, nitrate, chlorophyll, particulates, and salinity data from Monterey Bay. The emphasis of the project was on adaptive sampling so as to provide data for ROMS forecasting models.

3.2 AUV path planning

Research in path planning, localization, navigation, dynamics, and kinematics for AUVs is not uncommon. Path planning for AUVs is unique from path planning for traditional wheeled robots in that an AUV's environment - a body of water - is explicitly three-dimensional. The water environment in which they operate also introduces forces - i.e., currents - onto the robot that wheeled robots will almost never experience. Water also makes communication with a robot low bandwidth at the very least - offloading path planning coordination to a more powerful computer becomes a difficult task that terrestrial robots do not have to worry about. In short, path planning for AUVs is a significantly different problem from traditional methods. This section discusses approaches that take on the task of path planning for AUVs.

In his Ph.D thesis, Wang desires to collect ocean data in order to reduce underwater acoustic prediction uncertainties [29]. He proposes a solution to op-

timize "the location of in-situ measurements in an adaptive manner." Like the experiments in this thesis, an AUV follows a predetermined path created by searching a simplified graph in order to collect in-situ conductivity, temperature, and depth data. Additional work is done by allowing the AUV to adaptively generate its path on-board using Dynamic Programming.

In [23], Pêtrès et al. acknowledge the low bandwidth and current-prone situation of an AUV. They present an algorithm called Fast-Marching* that can *efficiently* extract a continuous 2-D path from a discrete, gridded environment. This algorithm takes advantage of the efficiency of A* search and the accuracy of an image processing algorithm to discover a continuous path in a discretized world. They also discuss a multiresolution solution in which the explored region is triangulated.

Lastly, there are a number of algorithms that can work or be extended to work independent of the medium in which the robot operates. Alvarez et al. [2] discuss the use of genetic algorithms to find low-energy paths for AUVs. The Focussed D* algorithm [26] allows for dynamic (in-route) replanning as more information about the environment is gathered. Multiresolution grids [9], potential fields [17], and probabalistic methods have also been used for robot path planning [25].

Chapter 4

Problem statement

The goal of this thesis is to design a ROMS-based ocean modeling system that intelligently plans for and assimilates AUV measurements in attempts to minimize the standard deviation of the assimilated ROMS model.

4.1 Problem definition

The problem - the minimization of standard deviation - and the formal definitions of the problem space are illustrated in detail in this section.

Let grid L be the three-dimensional lattice of nodes spanning the coastal area of interest:

$$L = \{\chi_{i,j,k} \mid i \,\epsilon\, [0, I],\ j \,\epsilon\, [0, J],\ k \,\epsilon\, [0, K]\} \tag{4.1}$$

Each node $\chi_{i,j,k}$ in the lattice is specified by its 3-D location within the georefer-

enced global coordinate frame:

$$\chi_{i,j,k} = [x_{i,j,k},\ y_{i,j,k},\ z_{i,j,k}] \tag{4.2}$$

where x is longitude, y is latitude, and z is depth.

The model M is defined as the ocean parameter estimates for each node in the lattice grid L:

$$M = \{m_{i,j,k} |\ i\ \epsilon\ [0, I],\ j\ \epsilon\ [0, J],\ k\ \epsilon\ [0, K]\} \tag{4.3}$$

$$m_{i,j,k} = [\chi_{i,j,k},\ S_{i,j,k},\ \sigma_S,\ T_{i,j,k},\ \sigma_T,\ u_{i,j,k},\ \sigma_u,\ v_{i,j,k},\ \sigma_v] \tag{4.4}$$

where χ is the three-dimensional point in Equation 4.2, and S, T, u, and v are the salinity, temperature, and latitudinal and longitudinal current velocities at χ, respectively. The σ values represent the standard deviation of each S, T, u, or v value at χ.

The cost function is then:

$$J_L(\chi) = \sum_{\chi_i\ \epsilon\ L} \sigma_T(\chi_i) \tag{4.5}$$

That is, J_L is the sum of the temperature standard deviation over all points in the domain. This problem can be formally written as:

$$\min_M\ J_L(\chi) \tag{4.6}$$

Chapter 5

Proposed solution

Standard deviation is one indicator of uncertainty. In this thesis, it is hypothesized that the cost function in Equation 4.5 above can be minimized by collecting temperature data at points at or near the surface where standard deviation is highest in the original model. Further, gathering temperature data from areas of higher standard deviation in the original model will most reduce the standard deviation in the assimilated model. The model mentioned hereafter has been reduced to a two-dimensional horizontal cross-section of the original model taken at the surface (so as to prevent the need for the Iver2 to dive to unsafe depths as well as making paths easier to develop). A system to address and test this solution is proposed here. A formal definition of the proposed solution is then described. Finally, an algorithm used to create optimal paths to maximize the acquisition of data in high temperature standard deviation locations is presented.

5.1 System diagram

An illustration of the proposed system is seen below in Figure 5.1.

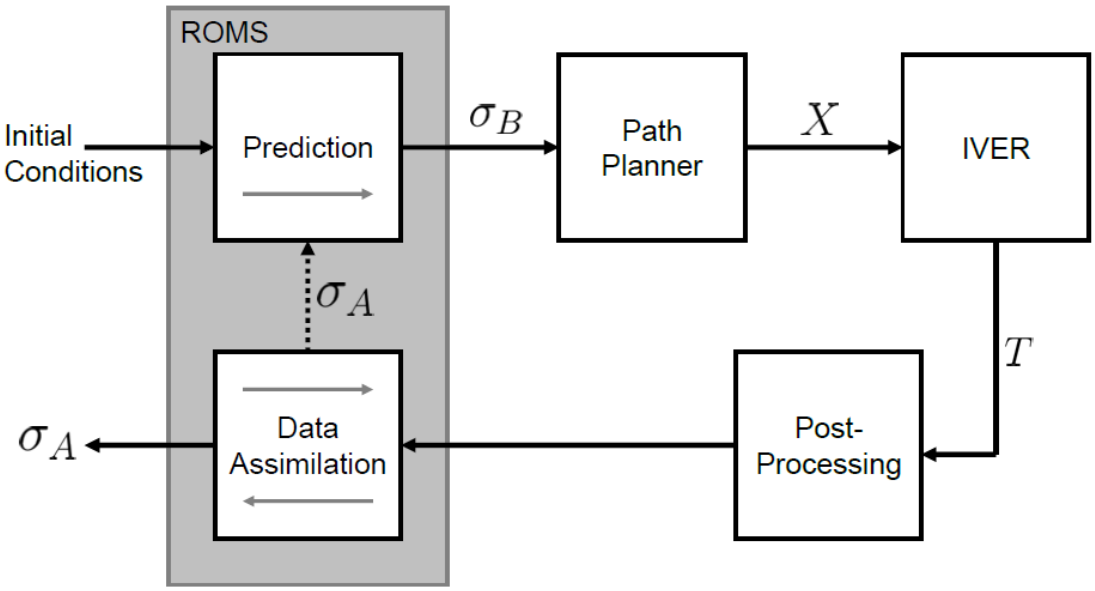

Figure 5.1: System diagram

In the first step of the system, initial conditions are introduced into ROMS' prediction component. Estimates of the resulting ocean model's standard deviations (σ_B, or *background*) are given to a path planner (described in Section 5.3 below). An AUV path is produced (X) and downloaded to the Iver2 AUV. The AUV is then deployed to gather temperature data (T) along the waypoints of that path. The temperature data is then post-processed so that it can be assimilated into the ROMS model. Finally, a new estimation of the the model's standard deviation is produced (σ_A, or *analysis*). Ideally, σ_A will be smaller than σ_B, indicating a reduction in standard deviation. To note, σ_A can be fed into ROMS prediction component (as shown in Figure 5.1) and the entire procedure repeated.

5.2 Solution definition

The goal of reducing the standard deviation of a model M by minimizing the cost function J_L is thought to be accomplished by gathering data where the standard deviation is greatest. This is formally described here, using nomenclature from the problem statement above.

The AUV's trajectory is defined by the sequence of lattice points visited along its path of length n:

$$X = [\chi^0 \, \chi^1 \, \chi^2 \, \cdots \, \chi^n] \tag{5.1}$$

The problem is to construct a trajectory that visits the lattice points of L, where temperature standard deviation, σ_T, is greatest in M. Hence the cost function to "maximize" is:

$$J(X) = \sum_{i=0}^{n} \sigma_T(\chi^n) \tag{5.2}$$

This problem can be formally written as:

$$\max_X \; J(X) \tag{5.3}$$

subject to:

$$\forall \, i \, \epsilon \, \chi_{i,j,k} : \; i = 0 \tag{5.4}$$

$$X^l \, \epsilon \, L \qquad \forall \, l \, \epsilon \, [0, n] \tag{5.5}$$

$$|X^t - X^{t-1}| = W_m \tag{5.6}$$

where $| \bullet |$ is the Euclidean distance function, and W_m is the constant distance between consecutive lattice nodes in L.

More succinctly, the goal is to maximize $J(X)$ by searching model M to identify a path X *on the surface* ($\mathrm{i} = 0$) that contains consecutive adjacent points $\chi^0_{i,j,k} \cdots \chi^n_{i,j,k}$ whose sum of $\sigma_T(\chi^t)$ (temperature standard deviation) is greater than any other path X.

5.3 Path planner

The Iver2 has limited computing capabilities and it is not computationally realistic to do an exhaustive search on a high-resolution (128 x 128) grid for long range paths. However, if the resolution of the model is reduced below a certain threshold, an exhaustive search becomes trivial.

A hybrid algorithm that combines model resolution-reduction techniques and a standard breadth-first algorithm was developed and used to plan paths for the Iver2. The algorithm was developed with the idea that it could eventually be adapted for use on the robot and utilized in real-time.

The path planning algorithm addresses the problem statement (Equations 4.5 and 4.6) by looking at *every* path returned by the breadth-first algorithm, thereby ensuring that the path with the greatest $J(X)$ (maximizing 5.2) is found.

The resolution reduction algorithm is introduced in Section 5.3.1, and the custom breadth-first algorithm in Section 5.3.2. They are combined into the path planning algorithm in Section 5.3.3. Visual examples of the resolution reduction are given in Section 5.3.4.

5.3.1 Resolution-reduction algorithm

This algorithm takes a square, two-dimensional model M_1 whose sides are of length 2^n and returns another square model M_2 whose sides are of length 2^{n-1}. The data in each adjacent 2×2 square in the original model is averaged and assigned to the respective grid cell in the lower resolution model. A pseudocode implementation of the algorithm can be found in Algorithm 1 below.

Algorithm 1 *ReduceResolution*(*model*)

Require: A model M_1 whose dimensions are a power of 2

```
M_2 = An empty (M_1.edgeLength/2 × M_1.edgeLength/2) array
for newRowIndex in M_2.edgeLength do
    for newColIndex in M_2.edgeLength do
        newVal = 0
        for oldRowIndex in range(newRowIndex, newRowIndex+1) do
            for oldColIndex in range(newColIndex, newColIndex+1) do
                newVal+ = M_1[oldRowIndex][oldColIndex]
            end for
        end for
        M_2[newRowIndex][newColIndex] = newVal/4
    end for
end for
return M_2
```

5.3.2 Breadth-first algorithm

The algorithm described in Algorithm 2 is an adaptation of the classical breadth-first search algorithm [19, 35]. The algorithm searches any 2-D model and returns all paths connecting the start and end points, given a distance constraint. This algorithm forces the distance contraint to be exhausted - that is, no paths between the start and end points that are shorter (or longer) than the maximum distance are accepted. The algorithm will return without any paths if the search space is too large.

Algorithm 2 $BreadthFirstSearch(model, start, end, maxDistance)$

Require: *start* and *end* points whose coordinates are within model M_1
Require: A 2-D model M_1
Require: A *maxDistance* that the robot can travel

```
1: GRID_PT_RES = distance between any two adjacent points
2: paths = empty List() {a list to hold all final paths}
3: queue = empty Queue()
4: startNode = Node(start, distLeft = maxDistance, parent = None)
5: queue.push(startNode)
6: while queue not empty do
7:     node = queue.next()
8:     if node.numParents > MAX_NUM_PARENTS then
9:         return  {search space is too large (grid resolution too high)}
10:    end if
11:    adjacentPoints = getAdjacentPoints(node, M_1)
12:    for legalPt in adjacentPoints do
13:        if legalPt ≠ end and node.pathContains(legalPt) then
14:            continue
15:        end if
16:        cNode = newNode(legalPt) {create a child node}
17:        cNode.error = node.error + errorAt(legalPt)
18:        cNode.distLeft = node.distLeft - distBtwn(legalPt, node.pt)
19:        cNode.parent = node
20:        if legalPt == end then
21:            if (cNode.distTo − maxDistance) ≤ GRID_PT_RES then
22:                paths.append(cNode)
23:            end if
24:        else {can still visit more points}
25:            queue.push(cNode)
26:        end if
27:    end for
28: end while
29: return paths
```

5.3.3 Path planning algorithm

The two supporting algorithms in Sections 5.3.1 and 5.3.2 above are combined in this algorithm to produce a hybrid algorithm that allows for rapid, optimal path planning.

Algorithm 3 $PathPlanningAlgorithm(start, end, model, maxDistance)$

Require: *start* and *end* points whose coordinates are within model M_1
Require: A model M_1 whose dimensions are a power of 2
Require: A $maxDistance$ that the robot can travel

```
origModel = M_1 {keep a copy of the original model}
while M_1 cannot be searched exhaustively do
    M_1 = ReduceResolution(M_1)
end while
allPaths = BreadthFirstSearch(M_1, start, end, maxDistance)
bestPath = findBestPath(allPaths)
{increase the model resolution until a path is found from start to end at the highest resolution}
while M_1 not at highest resolution do
    M_1 = origModel
    while a higher-res path can't be found between points in bestPath do
        M_1 = ReduceResolution(M_1)
    end while
    bestHigherResPath = empty List()
    for each pt in bestPath do
        maxDistance = distBtwn(pt, nextPt)
        allPaths = BreadthFirstSearch(M_1, pt, nextPt, maxDistance)
        bestHigherResPath.append(findBestPath(allPaths))
    end for
    bestPath = bestHigherResPath
end while
return bestPath
```

This algorithm takes as a parameter a square 2-D, high resolution model, start and end points, and a maximum distance that the robot can travel. The model is reduced in resolution until it can be searched quickly by a breadth-first algorithm. Once reduced appropriately, a number of low-resolution paths are found in seconds. A simple function identifies the best of these paths by choosing

the one that has maximized the cost function $J(X)$, as in Equation 4.5.

The algorithm then builds the final path by creating higher-resolution paths between adjacent lower-resolution points. The above process is essentially repeated with a smaller *maximum distance* until a path has been found at the highest resolution. The *maxDistance* assignment above can be increased by a multiplier to allow for "play" in the path creation between the lower resolution points.

5.3.4 Examples

The figures below are visualizations of the model that the path planning algorithm "sees" at various resolutions. Figure 5.2 shows the model at its highest resolution - a 128 x 128 model. A full breadth-first search of this model would take a significant amount of time on a typical desktop computer. Figures 5.3 and 5.4 show 32- and 16- square models, respectively. These models are searched much more easily.

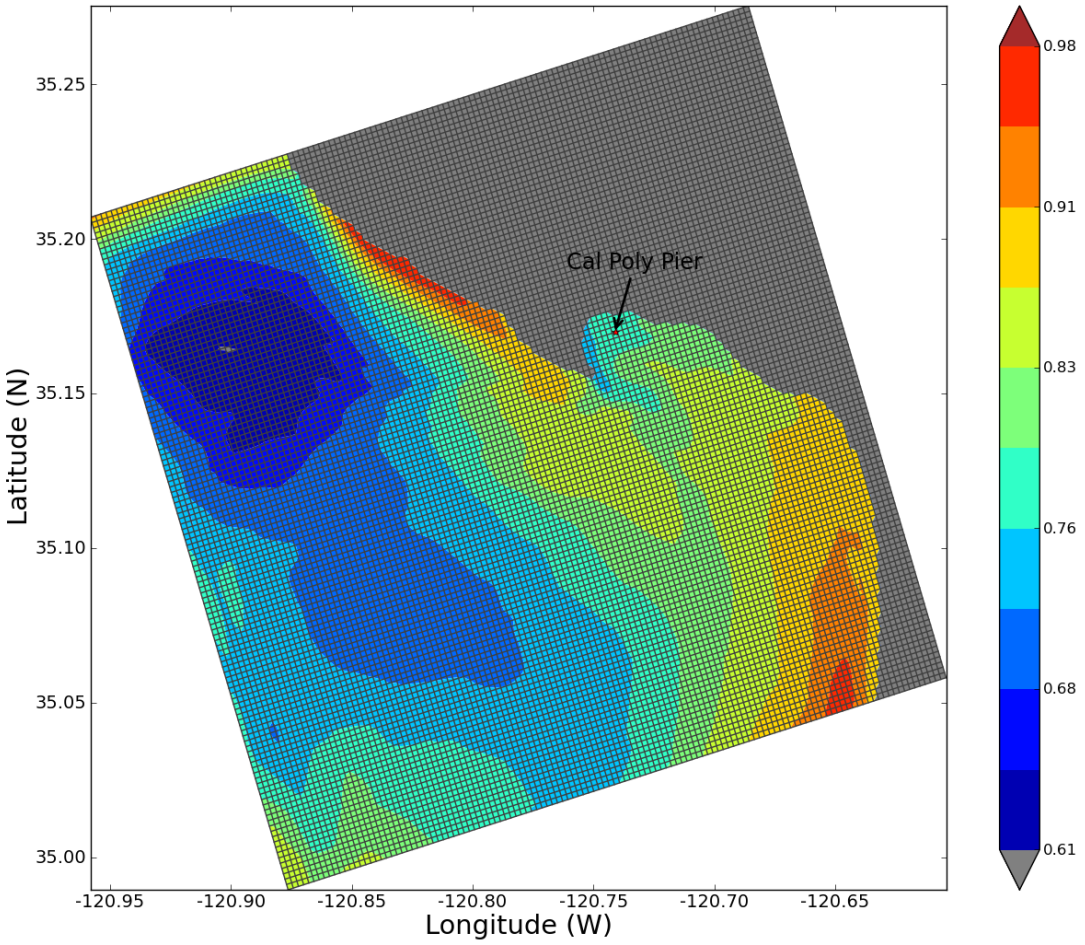

Figure 5.2: High resolution ROMS model - 128 x 128

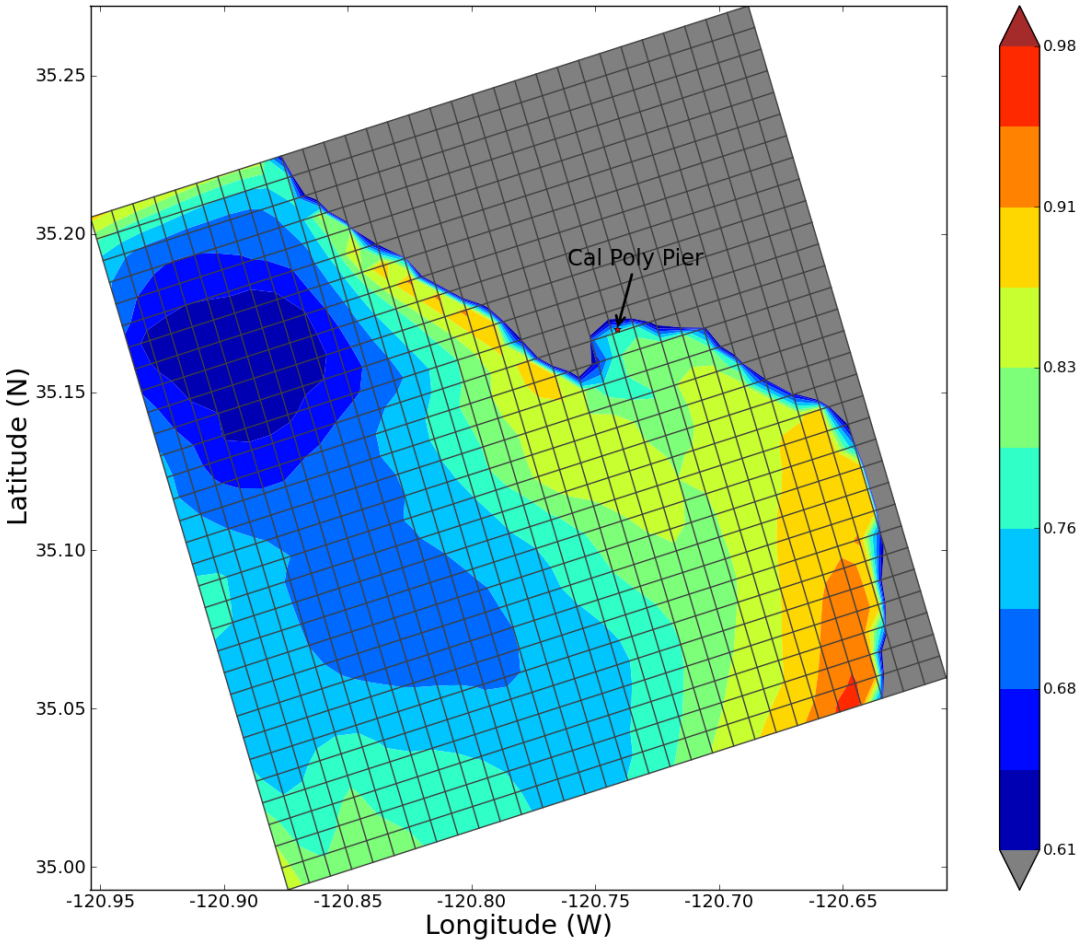

Figure 5.3: Lower resolution ROMS model - 32 x 32

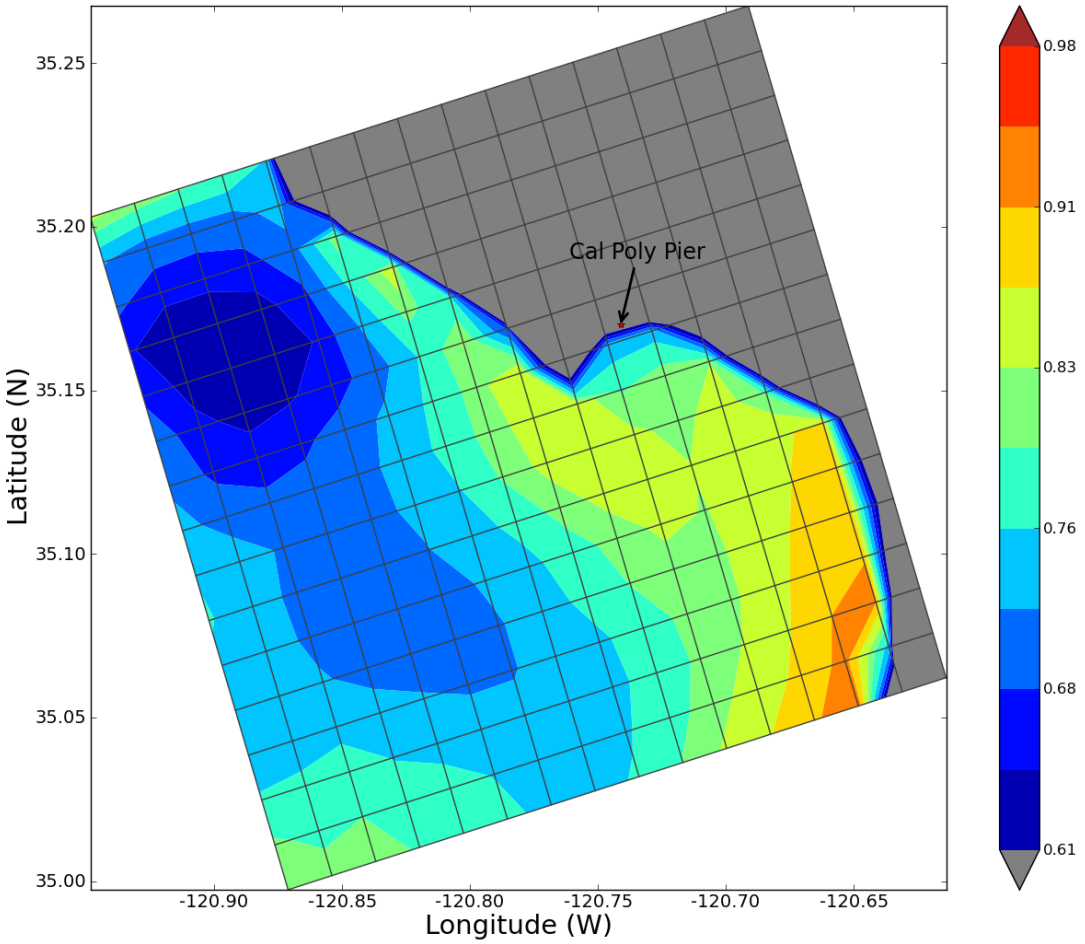

Figure 5.4: Lowest resolution ROMS model - 16 x 16

Chapter 6

Implementation

A path planning algorithm was developed as part of a solution to the problem statement. Though this algorithm is critical in exploring the differences between optimized and unoptimized missions, it cannot act alone. This section discusses the implementation details of the experiment and system, including the hardware and software components necessary for its completion.

6.1 Components

There were a number of hardware and software components critical to the completion of this experiment. The Iver2 AUV and attached temperature sensor were used to gather the *in situ* temperature measurements used as inputs to the ROMS models; a laptop computer and mobile wireless access point (a router) were required to communicate with the Iver2. The path planning algorithm and visualizations thereof were developed in Python, and proprietary OceanServer software was used to create and deploy these missions. ROMS was used to assimilate the temperature data and produce a model that can be used by the path

planning algorithm. This section discusses each component in more detail.

6.1.1 Iver2 AUV

The Iver2 AUV is the vehicle through which temperature measurements in San Luis Obispo Bay were taken. The Iver2 followed a pre-determined trajectory - made up of a number of GPS waypoints - created before each mission, gathering temperature measurements between and at each of these waypoints.

The Iver2 is designed and manufactured by OceanServer Technology, Inc. It is a a smaller (approximately 20 kg), 1-person-portable AUV made for rapid deployment and everyday use. It has a claimed battery life of 24 hours at 2.5 knots. The Iver2 is equipped with a number of integrated sensors (depth, pressure, compass, etc.) and can easily be fitted with additional sensors. For the experiments in this thesis, the Iver2 was equipped with a temperature sensor. An image of Cal Poly's Iver2 AUV can be seen in Figure 6.1.

The Iver2's processing power is supported by two embedded AMD Geode processors running on two different mini-ITX boards. The primary processor runs an installation of Windows XP Embedded specially suited for the Iver2's hardware, while the secondary processor runs a generic installation of Windows XP. The primary processor is only to be used for control and mission execution via OceanServer's Underwater Vehicle Console; user programs that access the Iver2's motor, fins, or sensors should be placed on the secondary processor. The temperature sensor was soldered to the serial port pins on the secondary processor so that the information could be accessed through a program on the secondary processor.

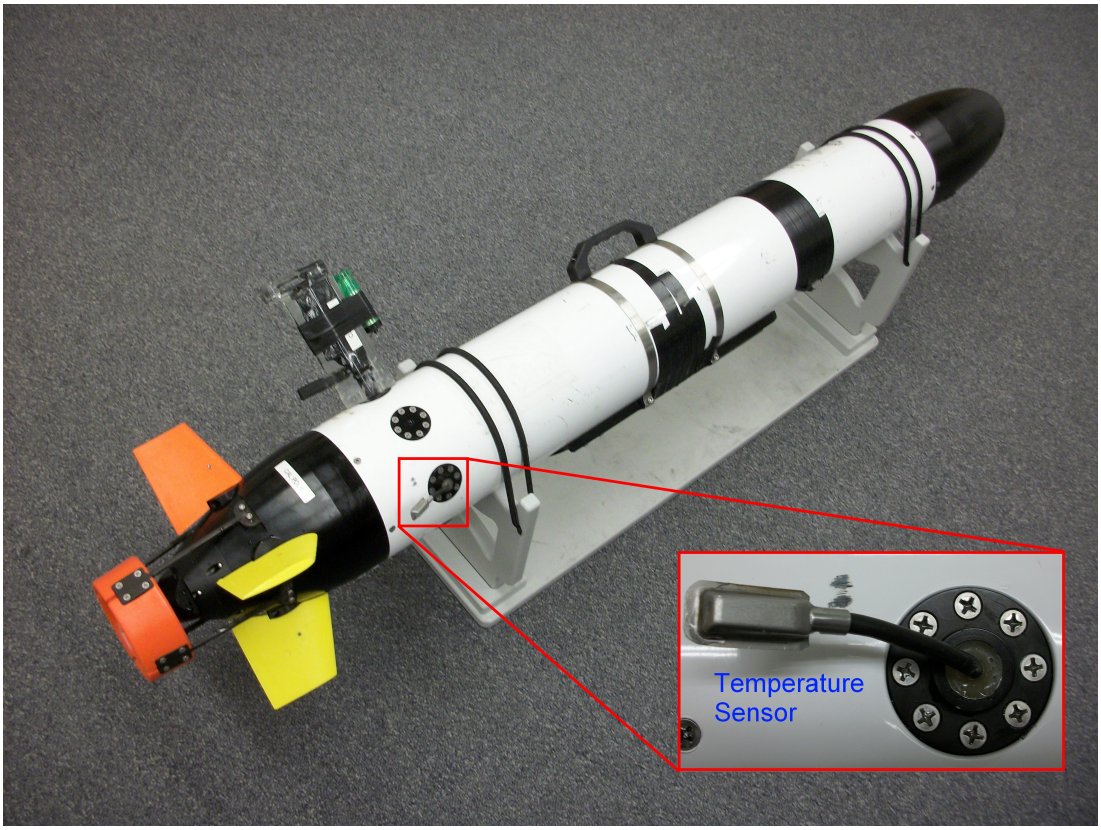

Figure 6.1: Cal Poly's Iver2 AUV

6.1.2 Temperature sensor

An Airmar T80 NMEA 0183 temperature sensor was attached to the Iver2 to gather temperature data. The small sensor unit sits on the outside of the Iver2 and connects to an interface box on the inside of the Iver2 that outputs NMEA temperature data. This data is transmitted via a serial link to the Iver2's secondary processor once per second at 4800 baud. The temperature sensor operates in waters between 0 and 30 degrees Celcius, is accurate to within 0.2 degrees [1], and has a resolution of .01 degrees. The temperature sensor is magnified in Figure 6.1 above.

6.1.3 Laptop and wireless access point

The laptop and access point are necessary to communicate with the Iver2, in or out of the water. The access point creates a wireless network called "IVER" to which the Iver2 and laptop connect over an 802.11g connection. The laptop then uses Windows' Remote Desktop to log in to the Iver2's desktop. Once logged in to the Iver2, a user can manually control the Iver2's direction and speed. The user can also copy a mission file that contains a collection of GPS coordinates, depths, and speeds desired for an autonomous mission; they can then send the Iver2 on an autonomous mission.

6.1.4 Python modules

The Python modules developed for this thesis provide MATLAB-like visualization of the ROMS models as well as the algorithms used to create mission trajectories (paths) for the Iver2. This was accomplished by reading ROMS data

into a two-dimensional Map object that can be easily searched using traditional graph search algorithms. See Section 5.3 for an in-depth description of the algorithms used for this experiment.

6.1.5 VectorMap software

VectorMap is a software program provided by OceanServer that provides a GUI through which a user can create missions for the Iver2 AUV. It allows the user to overlay GPS waypoints on a National Oceanic and Atmospheric Administration (NOAA) chart that contains bathymetric and topological data of the coastal area that will be travelled by the Iver2. VectorMap also allows the user to specify speed, depth, and heading of the Iver2 at each of the waypoints. Once the waypoints have been entered, VectorMap saves the information to a *.mis* file in a format that can be consumed by the Iver2's Underwater Vehicle Console, explained below. A screenshot of the VectorMap software can be seen in Figure 6.2.

6.1.6 Underwater Vehicle Console software

The Underwater Vehicle Console, or UVC, is an OceanServer program that operates on the Iver2's primary processing board. The UVC controls the motors and communicates with a variety of peripherals via serial communication. It has two modes - manual control and mission control. In manual control, the user can control the Iver2's speed and direction through a graphical interface. In mission mode, the Iver2 autonomously navigates to waypoints specified in a *.mis* file.

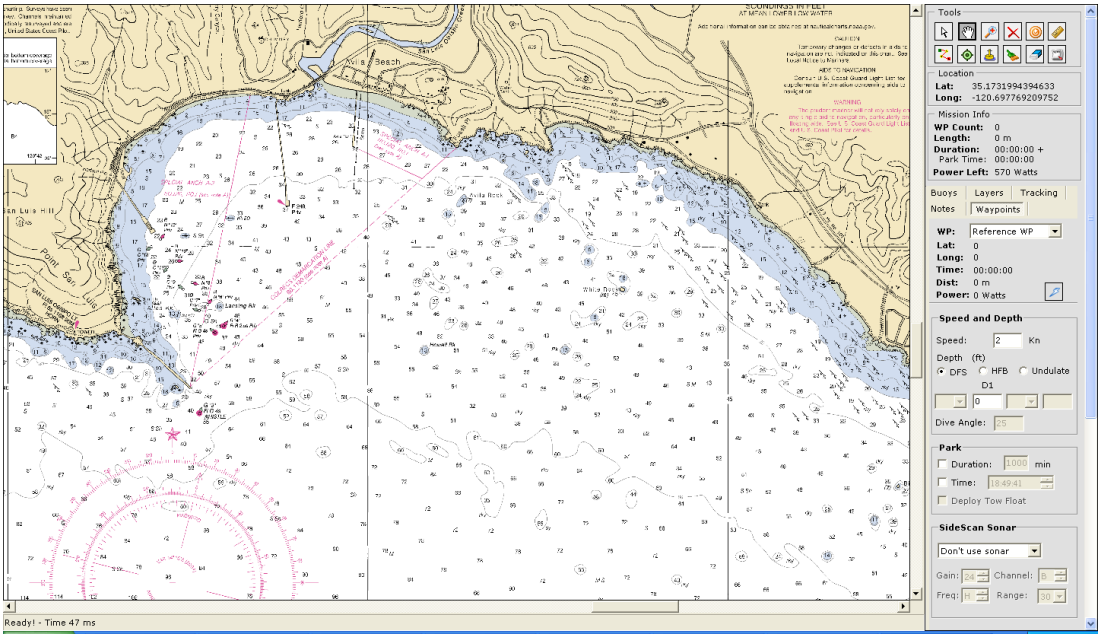

Figure 6.2: VectorMap mission planning software

6.1.7 ROMS San Luis Obispo Bay model

San Luis Obispo Bay, located about 11 miles southwest of Cal Poly, is the approximate center of the 29.2 x 38.4 kilometer (146 x 192 grid; points are 200m apart) ROMS model used in this experiment. The model was set up by Dr. Paul Choboter, a mathematics professor at Cal Poly and member of Cal Poly's Center for Coastal Marine Science. ROMS was run without data assimilation to simulate the state of the ocean for the time period of October and November 2009. It was configured to run on a rectangular area with coordinates (34.866407°N, 120.872460°W) at the bottom left corner of the model, and (35.271988°N, 120.684721°W) at the upper right corner. The model was forced with wind stress from NOAA/NCDC blended 6-hourly 0.25-degree sea surface wind stress (http://www.ncdc.noaa.gov/oa/rsad/blendedseawinds.html). Initial and boundary conditions were taken from HYCOM 1/12 degree

global hindcast reanalysis (http://www.hycom.org/, http://hycom.coaps.fsu.edu:8080/thredds/dodsC/glb_analysis.html).

That two month run was used to calculate the standard deviation of temperature field, shown in Figure 6.3 below. The temperature standard deviation was used by the path planning algorithm in Section 5.3 to develop paths that optimized the distance travelled. Temperature data acquired from both optimized and unoptimized missions were assimilated into the original model to produce new models that are discusssed in the results (Chapter 8).

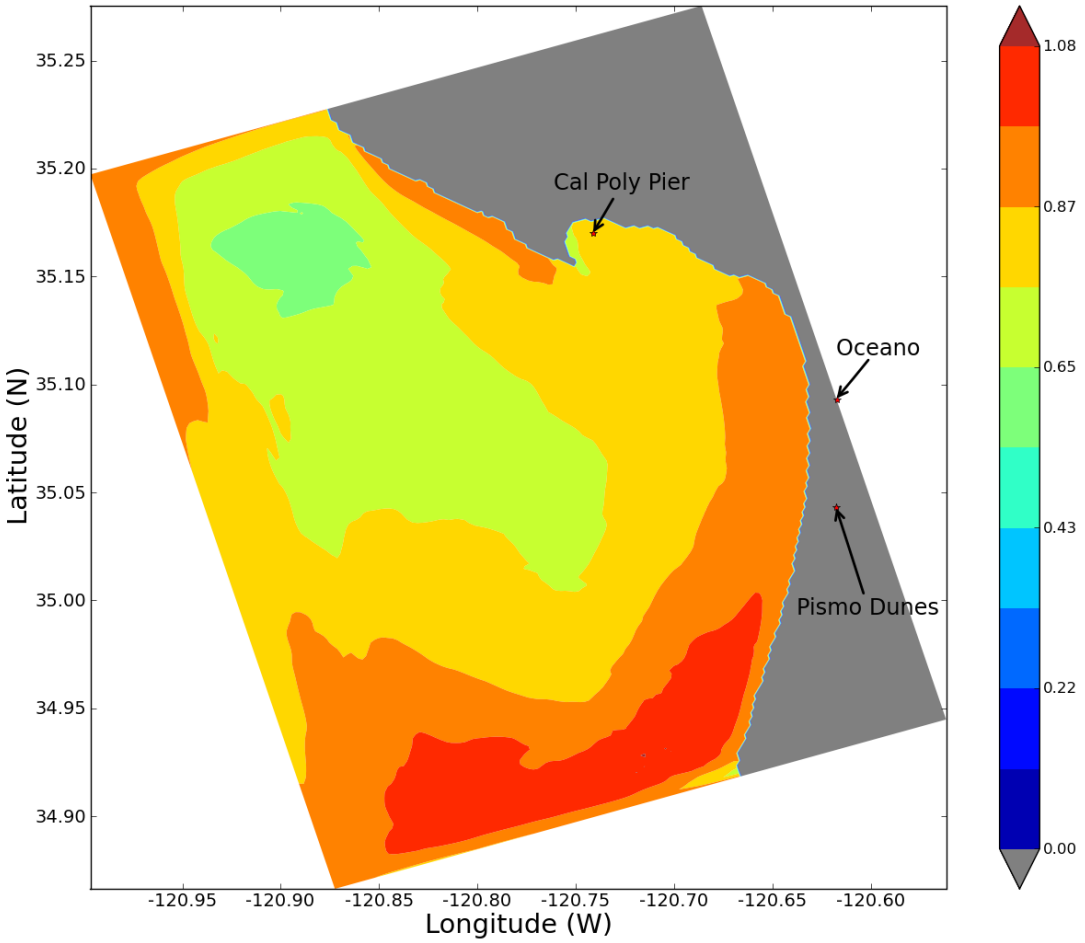

Figure 6.3: San Luis Obispo Bay model - Temperature standard deviation values from September - October 2009

Chapter 7

Experiments

The goal of the experiment was to use the Iver2 AUV and attached temperature sensor to gather temperature data in parts of San Luis Obispo Bay. The mission waypoints visited by the Iver2 were chosen at random in the first week (unoptimized missions), and by the path planning algorithm above in the second week (optimized missions). In both weeks, two shorter missions started and ended at the Cal Poly Pier, while a third, longer mission was launched from a boat off of the Pismo Beach dunes. This chapter describes the procedure used to develop optimized and unoptimized missions in order to test my problem statement. It discusses the nomenclature used throughout the rest of this paper and provides visualizations of the desired optimized and unoptimized missions.

7.1 Nomenclature and quick reference

For brevity, missions will be referred to as $O\#$ and $U\#$ for optimized and unoptimized missions, respectively. Table 7.1 below shows the date of each mission and the abbreviated name to which it will be referred.

	Date	Mission Abbreviation
Unoptimized	Nov 8	U1
	Nov 10	U2
	Nov 11	U3
Optimized	Nov 16	O1
	Nov 17	O2
	Nov 23	O3

Table 7.1: Mission name abbreviations

A quick comparison of the distance, depths, and speeds of the missions can be seen in Table 7.2 below:

Mission	Distance (m)	Max Depth (f)	Submerged speed (knots)	Surface speed (knots)
U1	11500	0	N/A	2.5
U2	23200	15	4	2.5
U3	12700	15	4	2.5
O1	13200	15	4	2.5
O2	19700	10	3.5	2.5
O3	25800	10	3.5	2.5

Table 7.2: Mission dates, lengths, depths, and speeds

7.2 Week 1: Unoptimized missions

The unoptimized missions were executed on November 8, 10, and 11, 2010 (U1, U2, U3, respectively). Mission U2 was started from the boat approximately 11.5 km south of the Cal Poly Pier. The other two missions were started and finished at the pier.

The missions were developed by choosing a random but safe (i.e., away from shore, known kelp beds, and shallow areas) set of points in MATLAB and then porting these waypoints to VectorMap. An image of the original temperature deviation model has been overlayed with the unoptimized missions in Figure 7.1 below:

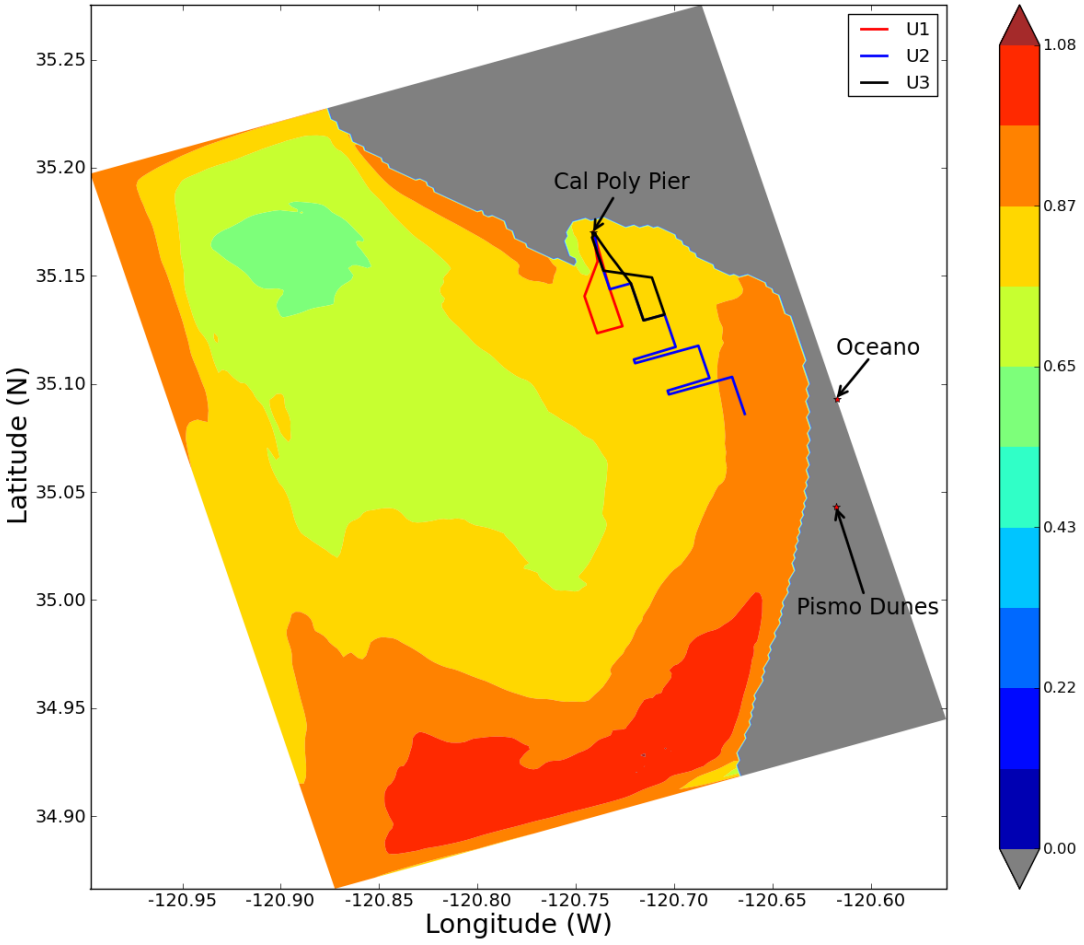

Figure 7.1: Unoptimized missions overlayed on the original temperature deviation model

7.3 Week 2: Optimized missions

The optimized missions were executed on November 16, 17, and 23, 2010 (O1, O2, O3 respectively). Mission O3 started from the boat near the start of U2. The other two were started and finished at the pier.

The optimized missions were developed by running the path planning algorithm in Section 5.3.3 on the 2-D standard deviation model in Figure 6.3 above. Three different *maximum distance* parameters were supplied to the algorithm so that the missions would not be too similar, just as the unoptimized missions were different.

An image of the original temperature deviation model has been overlayed with the optimized missions in Figure 7.2 below:

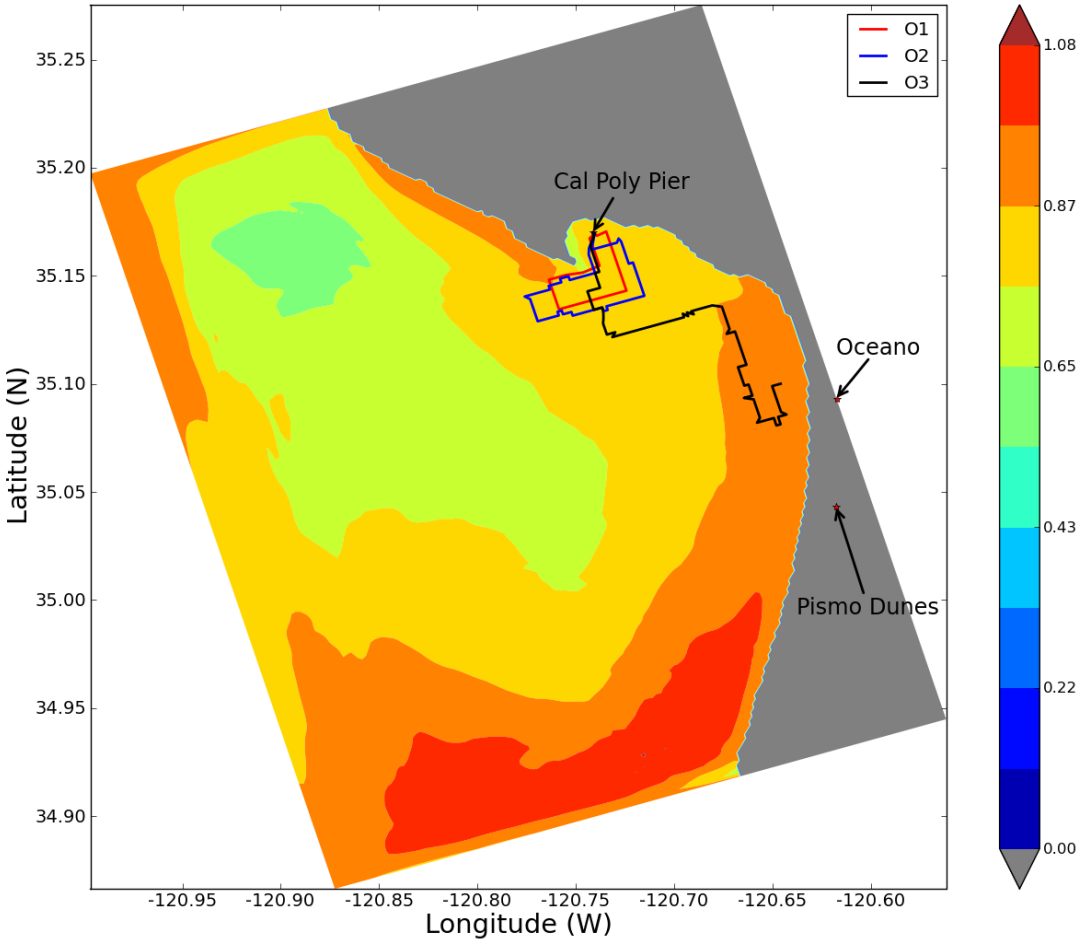

Figure 7.2: Optimized missions overlayed on the original temperature deviation model

Chapter 8

Results

Temperature data acquired from the experiments were grouped into a number of data points that were assimilated into ROMS to produce new San Luis Obispo Bay simulations with a new estimate of the temperature standard deviation.

For the purposes of the analysis, it is important that the results of the experiment are seen from a few different perspectives. As such, the results have been divided into a few different sections. Section 8.1 discusses the data assimilated simulations as they compare to the original model. Section 8.2 compares the results of optimized and unoptimized pier missions, while 8.3 compares the boat missions.

It should be noted that data from U2 has been compiled in three different ways (U2, U2.avg63, and U2.mid63) so as to allow for different comparisons against O3, which was cut short due to a mechanical failure on the Iver2.

8.1 Single-day effects of data on original model

The results of the experiment are detailed in the tables and figures below. Table 8.1 below compiles the most relevant data from each mission. The column names are explained after the table.

Mission	**Data Range (m)**	**Data Points**	$J(X)$	$J_L(\chi)$	$J_L(\chi)_{orig} - J_L(\chi)_{final}$	**% Reduction**
Original	-	-	-	18083	-	-
U1	8342	200	165	15041	3041	16.82%
U2	16293	276	233	14537	3546	19.61%
U2.avg63	15167	63	53	15010	3073	16.99%
U2.mid63	3287	63	53	15633	2450	13.55%
U3	11948	200	163	15115	2968	16.41%
O1	12488	200	164	14616	3467	19.17%
O2	13276	200	167	14503	3579	19.79%
O3	4439	200	58	16065	2018	11.16%

Table 8.1: Individual mission results

- **Mission** The identifier for the mission date, and whether it is optimized or unoptimized as in Table 7.1 above.
- **Data Range** The sum of the distances between each data point in the mission data. Some data from each mission was stripped at the beginning and/or end of each to facilitate ROMS processing. This value will be less than or equal to the planned mission lengths in Table 7.2.
- **Data Points** The number of data points in the *Data Range*; data from these points were assimilated into ROMS.
- $J(X)$ This value represents the sum of the standard deviation values at each *Data Point* from the original ROMS model as in Equation 5.2.

- $J_L(\chi)$ This value represents the sum of all non-land standard deviation values in the original or assimilated model as in Equation 4.5.

- $J_L(\chi)_{orig} - J_L(\chi)_{final}$ This value is the *difference* in total temperature standard deviation between the original model and the assimilated model. Larger values in this column are better.

- **% Reduction** This represents the percentage reduction in standard deviation from the original model. Equivalent to $(J_L(\chi)_{orig} - J_L(\chi)_{final})/J_L(\chi)_{orig}$.

U2.avg63 and U2.mid63 results are reductions of the U2 data made to allow for additional comparisons to O3. For U2.avg63, only 63 data points were taken across the length of the original U2, whose original 276 data points were averaged. This data was then assimilated to produce the results in the table. U2.mid63 consisted of data from 63 consecutive points from the middle of the original U2 data.

Figures 8.1, 8.2, 8.3, 8.4, and 8.5 show the effects of the unoptimized trajectories on the temperature standard deviation of the San Luis Obispo Bay model. Figures 8.6, 8.7, and 8.8 show the effects of the optimized trajectories on the temperature standard deviation in San Luis Obispo Bay. In all figures, the section of the mission from which the temperature data was actually used (*Data Range* in Table tab:results-single-day-error above) is overlayed on the model. Note that the standard deviation scale for these models are the same as in 6.3 above.

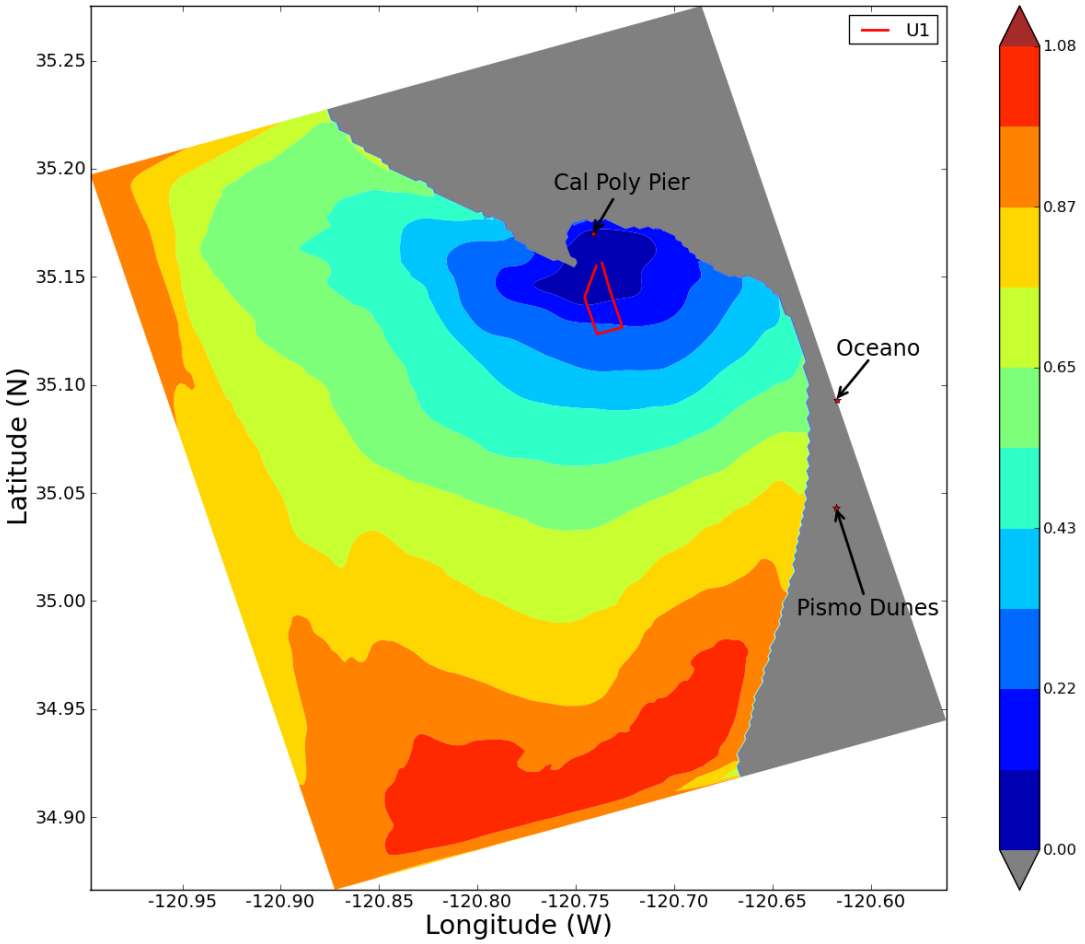

Figure 8.1: Temperature deviation after data assimilated from U1

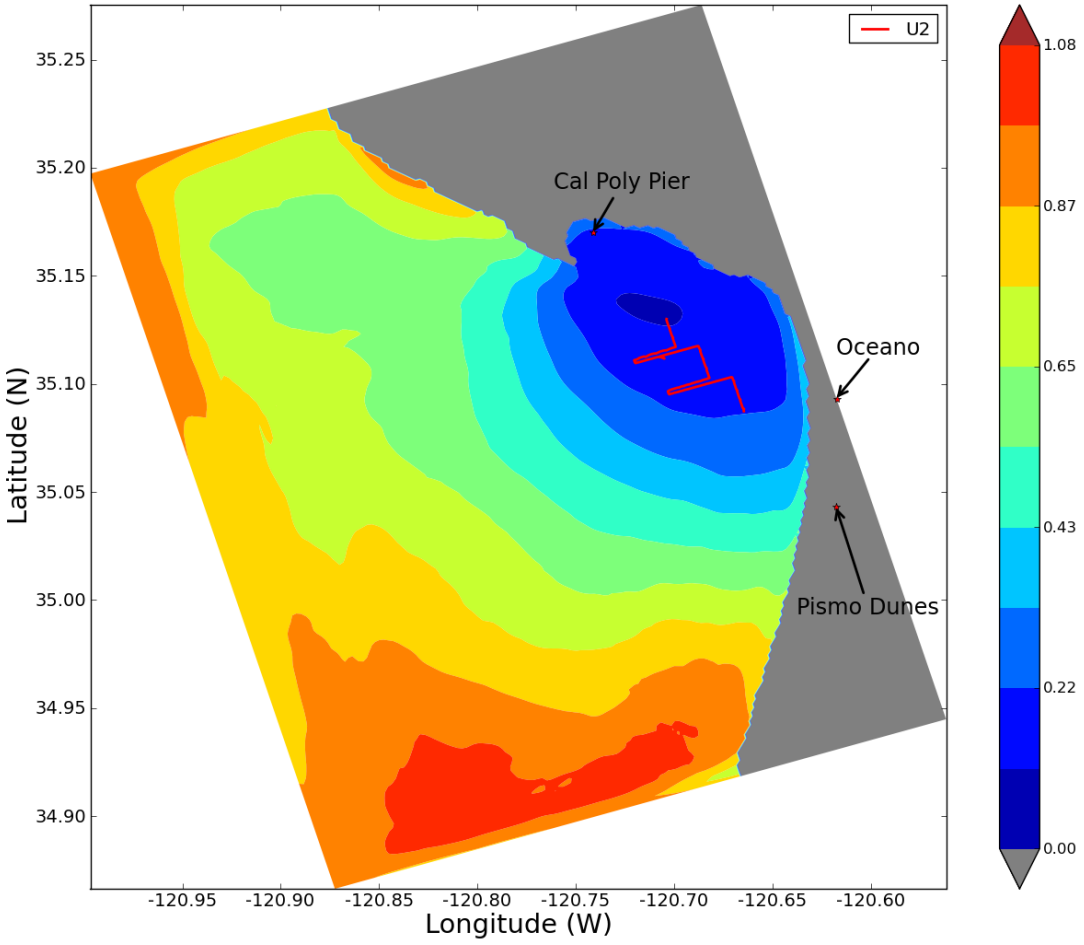

Figure 8.2: Temperature deviation after data assimilated from mission U2

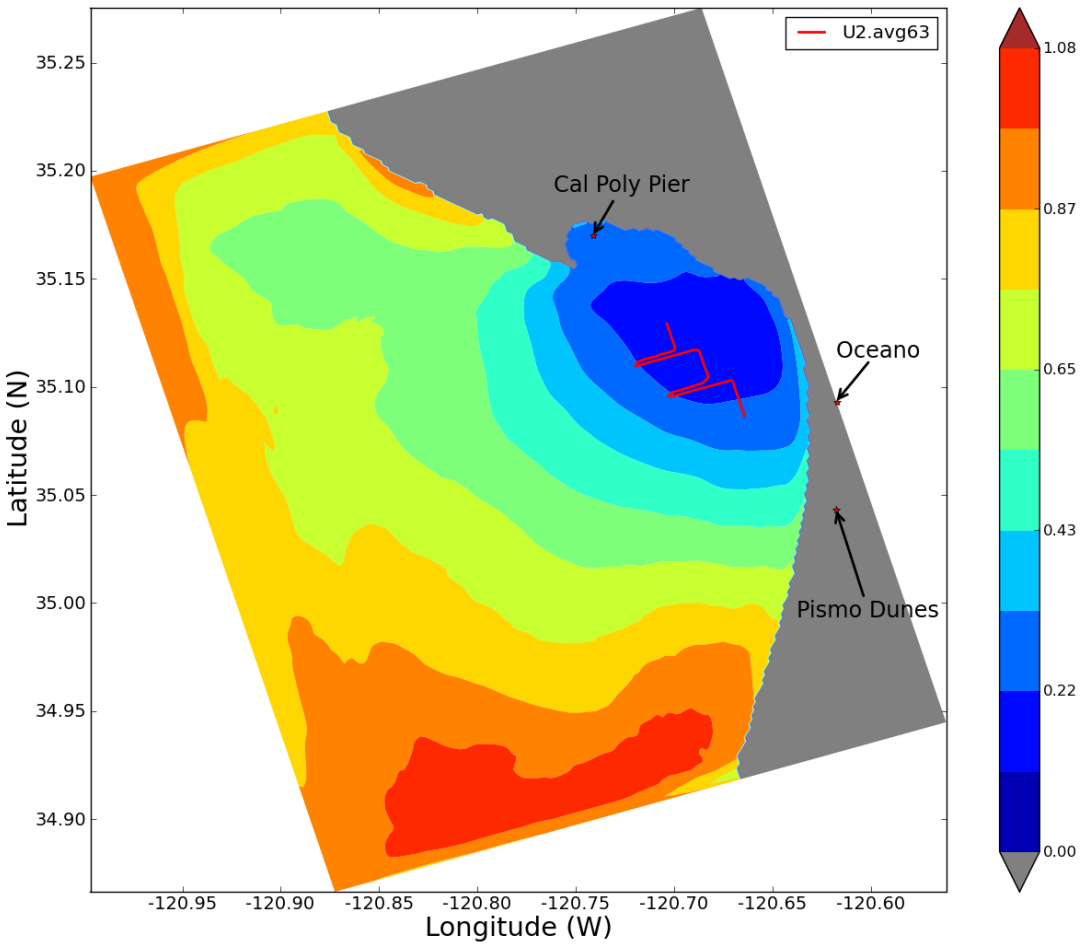

Figure 8.3: Temperature deviation after data assimilated from U2.avg63

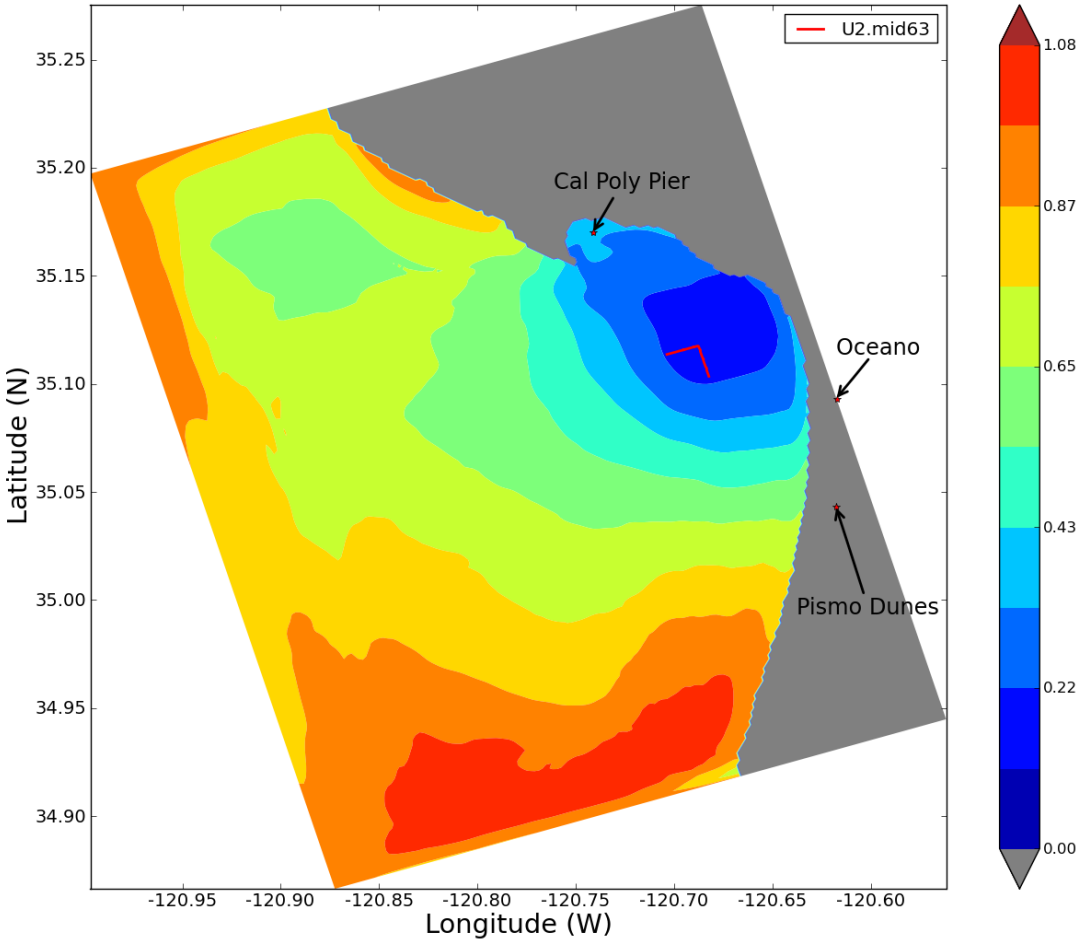

Figure 8.4: Temperature deviation after data assimilated from U2.mid63

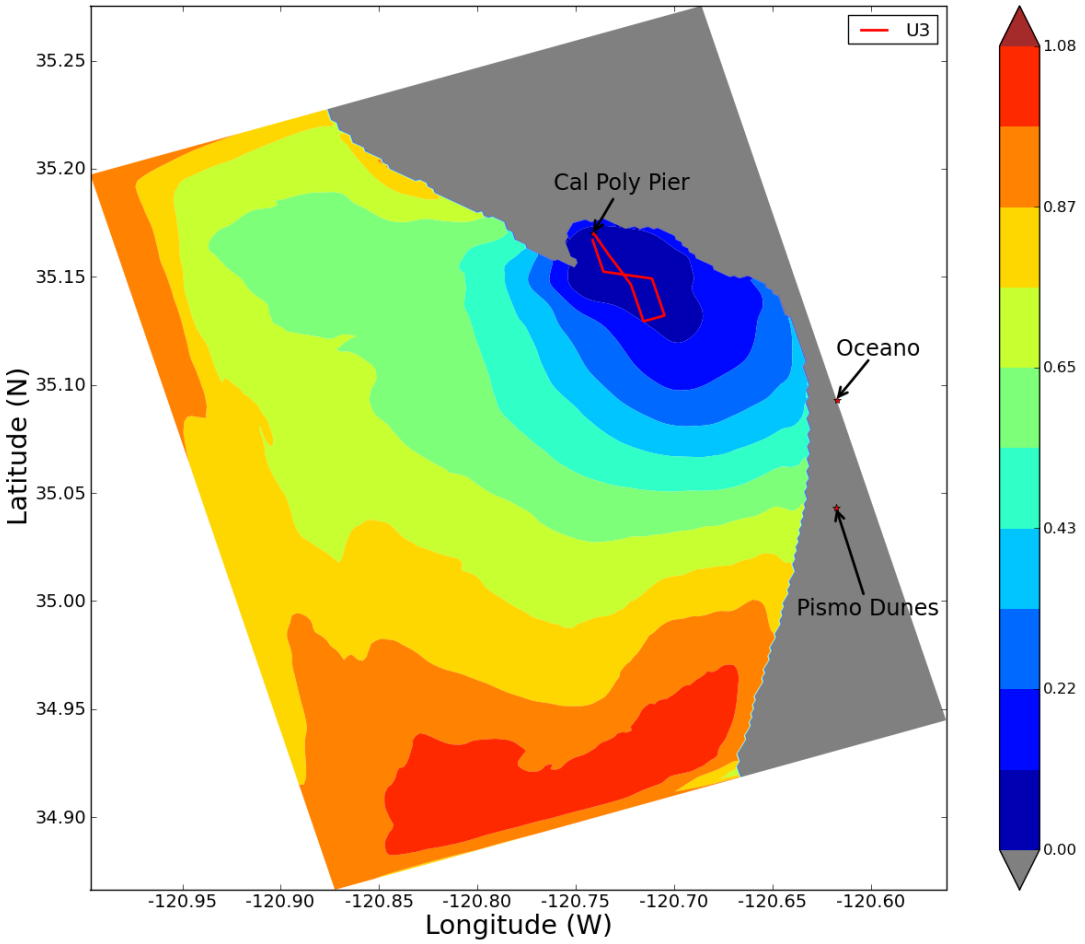

Figure 8.5: Temperature deviation after data assimilated from U3

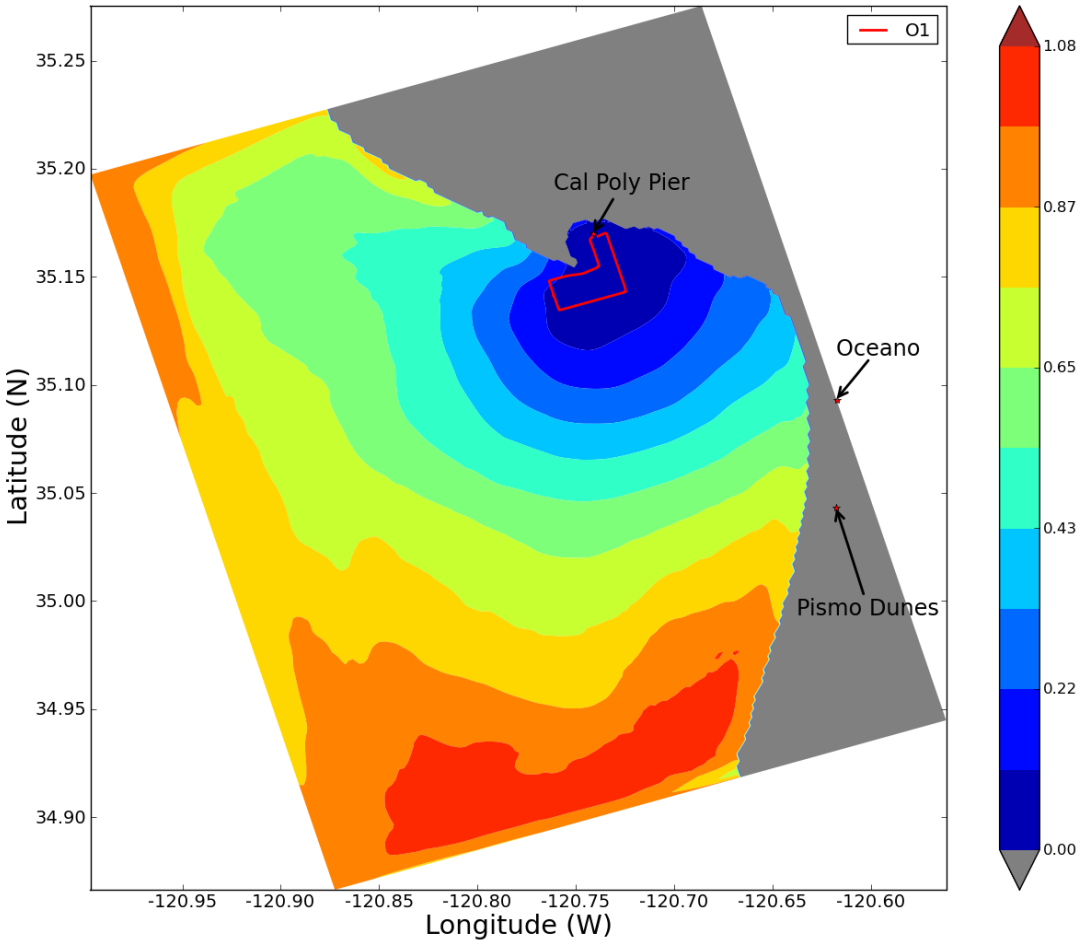

Figure 8.6: Temperature deviation after data assimilated from O1

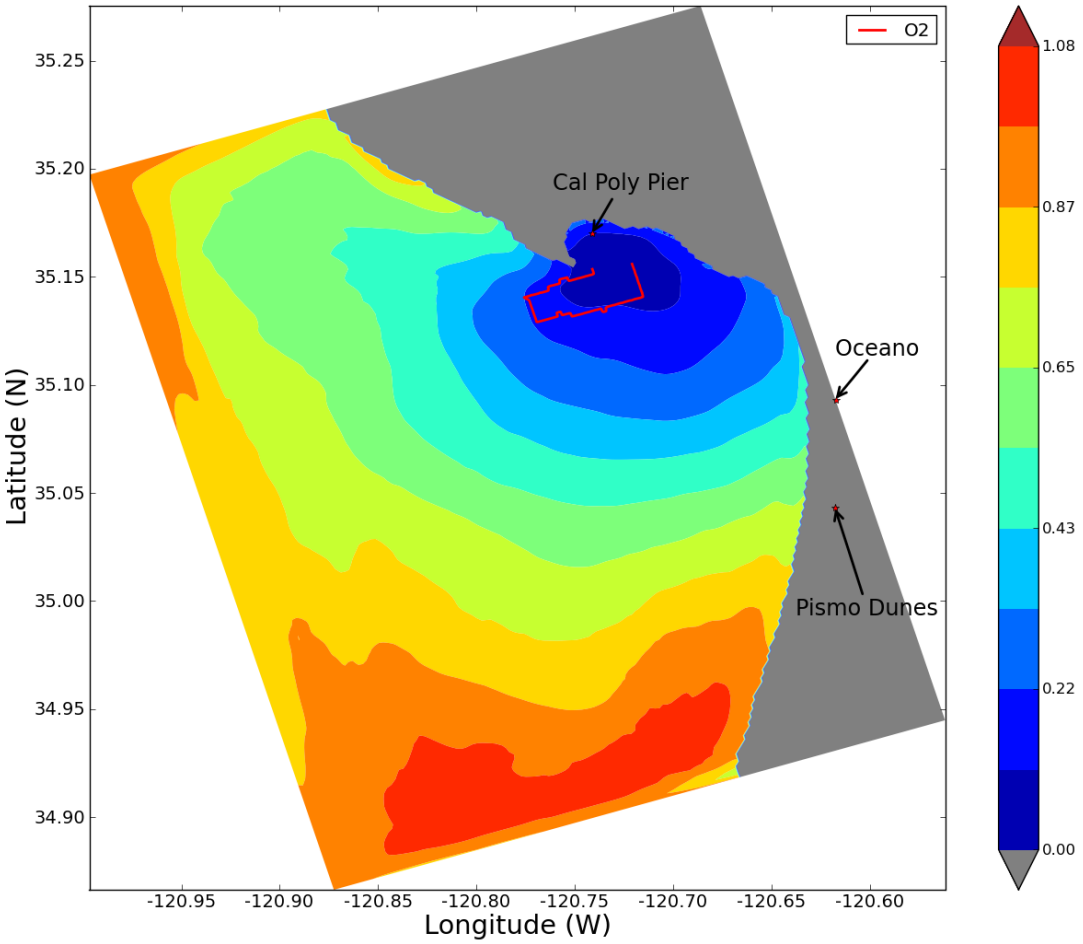

Figure 8.7: Temperature deviation after data assimilated from O2

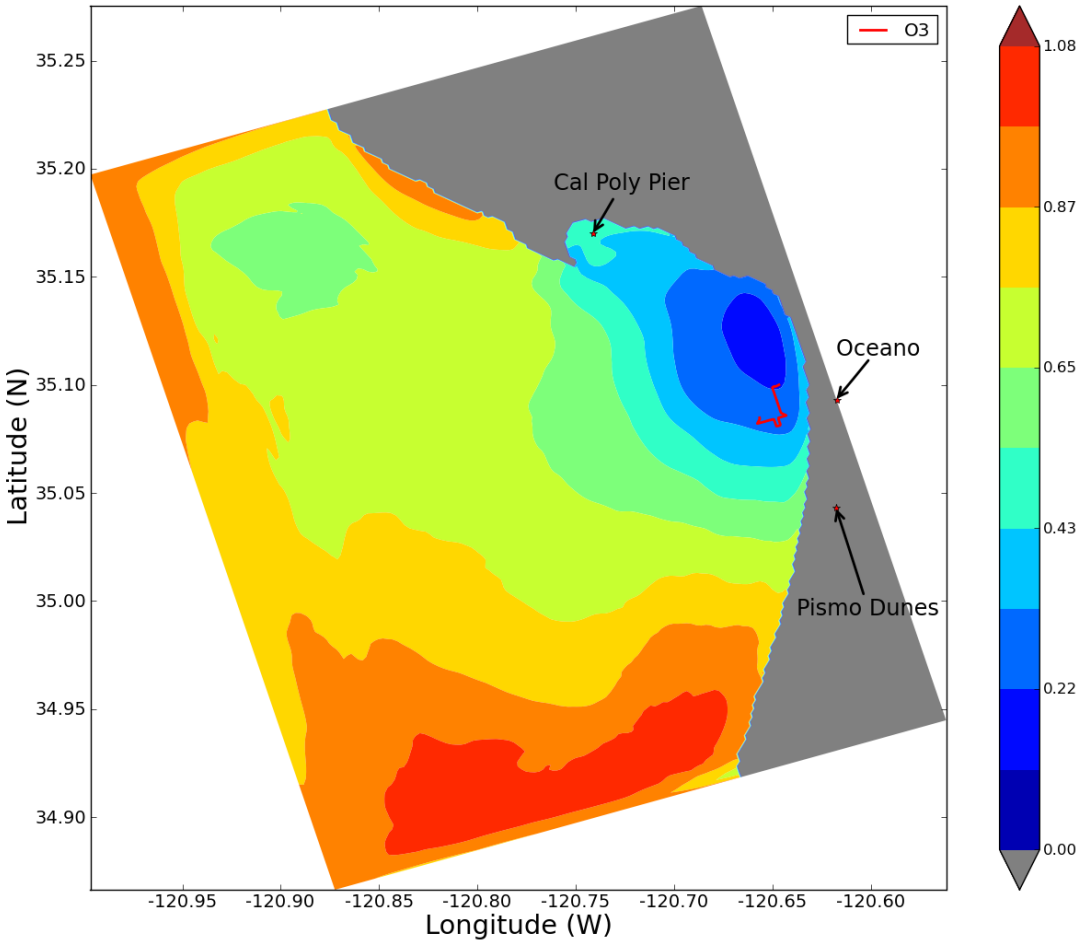

Figure 8.8: Temperature deviation after data assimilated from O3

The dark blue areas near the middle of each model show that the estimated standard deviation in that region is at or near zero. This means that if real temperature data were to be taken from that area in the time frame for which the model was made, it would be close or equal to the data in the model at the same data point.

8.1.1 Statistical results

Figure 8.9 below shows a scatter plot comparing the sum of standard deviation values ($J(X)$ in Table 8.1 above) along each mission from the original model to the percent reduction in standard deviation of the original model. The best-fit line, along with equation and correlation coefficient (r) are shown as well.

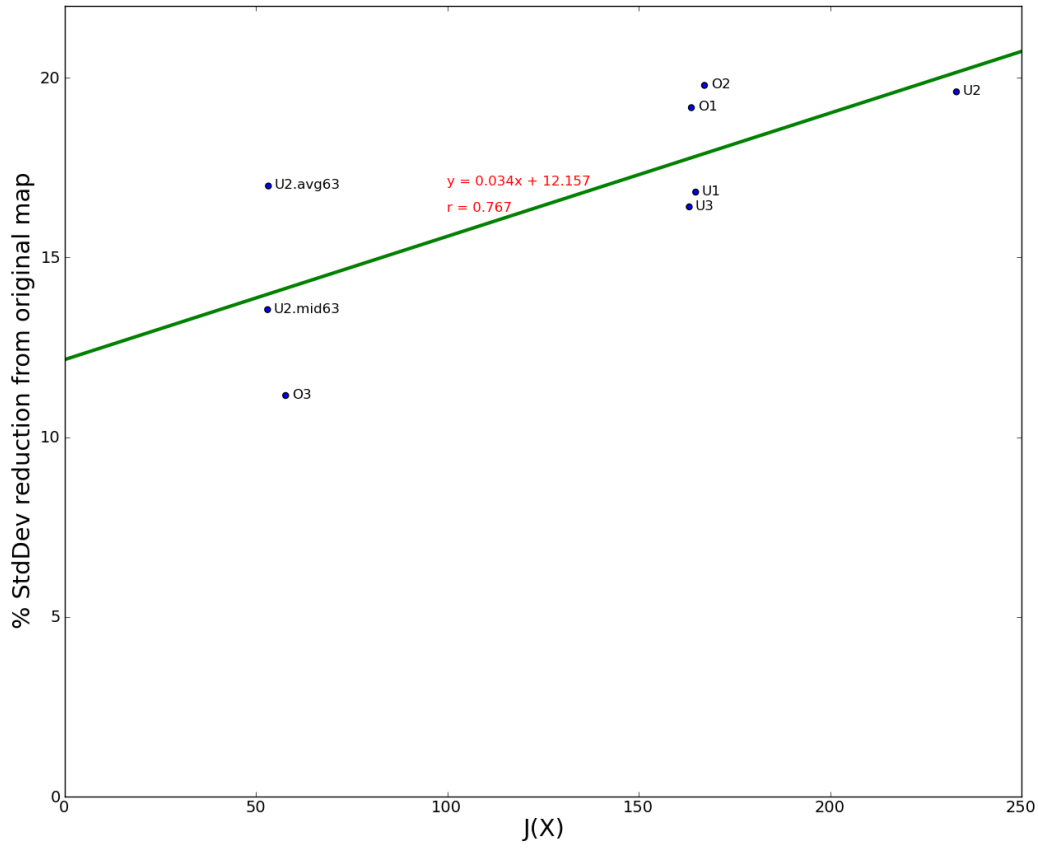

Figure 8.9: Scatter plot comparing the sum of standard deviation values along each mission path (x-axis) to the percent reduction in standard deviation of the original model (y-axis)

8.2 Comparison of missions launched from the pier

Visual and tabular results show that in both cases, the optimized missions from the pier decrease the model's standard deviation more than the unoptimized missions if measured over the entire model. However, the reduction was not uniformly better - in certain areas, the reduction in deviation from the unoptimized missions was greater than the optimized mission. Figures 8.10 and 8.11 below show these differences. In 8.10, the deviation model of U1 (Figure 8.1) was subtracted from O1 (Figure 8.6). The light-green-to-dark-blue colors show a negative difference - areas where the O1 model's deviation is smaller than that of U1. Likewise, the greenish-yellow-to-dark red colors identify places where deviation in the U1 model is smaller than O1. The black contour line simply offers a visual aid to identify the boundary between negative and positive values. Figure 8.11 is a comparison of O2 and U3; the color scale is the same as in 8.10.

Table 8.2 provides a numerical look at the data from Figures 8.10 and 8.11. The columns show the number of points where the standard deviation was reduced more by the optimized and unoptimized missions. These numbers are equivalent to the number of points inside or outside the contour lines in the figures below.

Missions	**# points where optimized > 0**	**# points where optimized < 0**	**Ratio**
U1 vs O1	16233	5522	2.94
U3 vs O2	19186	2585	7.42

Table 8.2: Deviation difference model comparisons

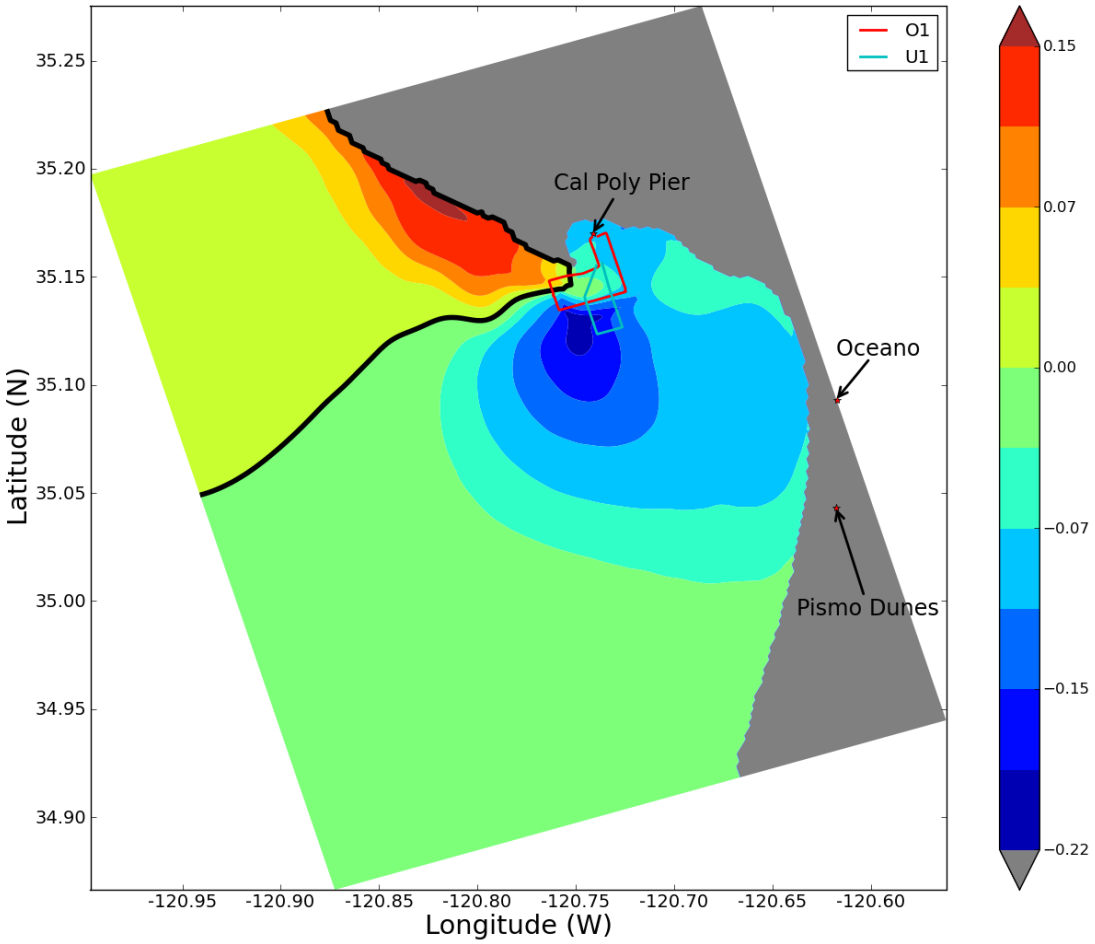

Figure 8.10: Standard deviation difference model - O1 vs U1

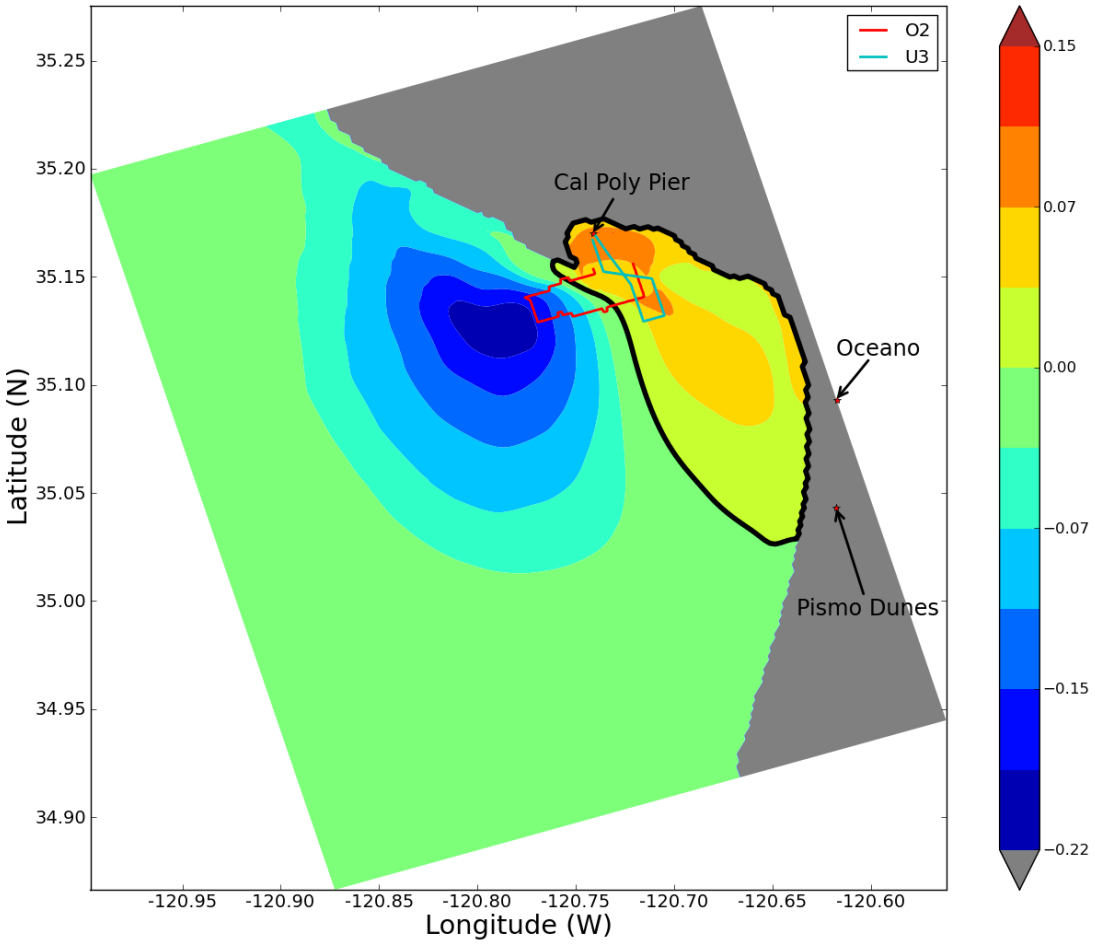

Figure 8.11: Standard deviation difference model - O2 vs U3

8.3 Comparison of missions launched from boat

Unlike the pier missions, it was the unoptimized mission that reduced the model's standard deviation more than its optimized counterpart. Unfortunately, the optimized mission completed less than 10% of the intended mission. Because of this, the unoptimized mission data was additionally manipulated such that closer comparisons could be made to the optimized mission in terms of mission length and number of data points used. Table 8.3 details these comparisons as Table 8.2 did for the pier missions above.

Missions	**# points where optimized > 0**	**# points where optimized < 0**	**Ratio**
U2 vs O3	27	21744	805.33
U2.avg63 vs O3	156	21615	138.56
U2.mid63 vs O3	4396	17375	3.95

Table 8.3: Comparison of missions started from boat

Figures 8.12, 8.13, and 8.14 show the standard deviation difference models between O3 and U2, U2.avg63, and U2.mid63, respectively. Like above, each of the assimilated models of U2 data are subtracted from the assimilated O3 model and plotted on the same scale. The black line indicates the zero-contour.

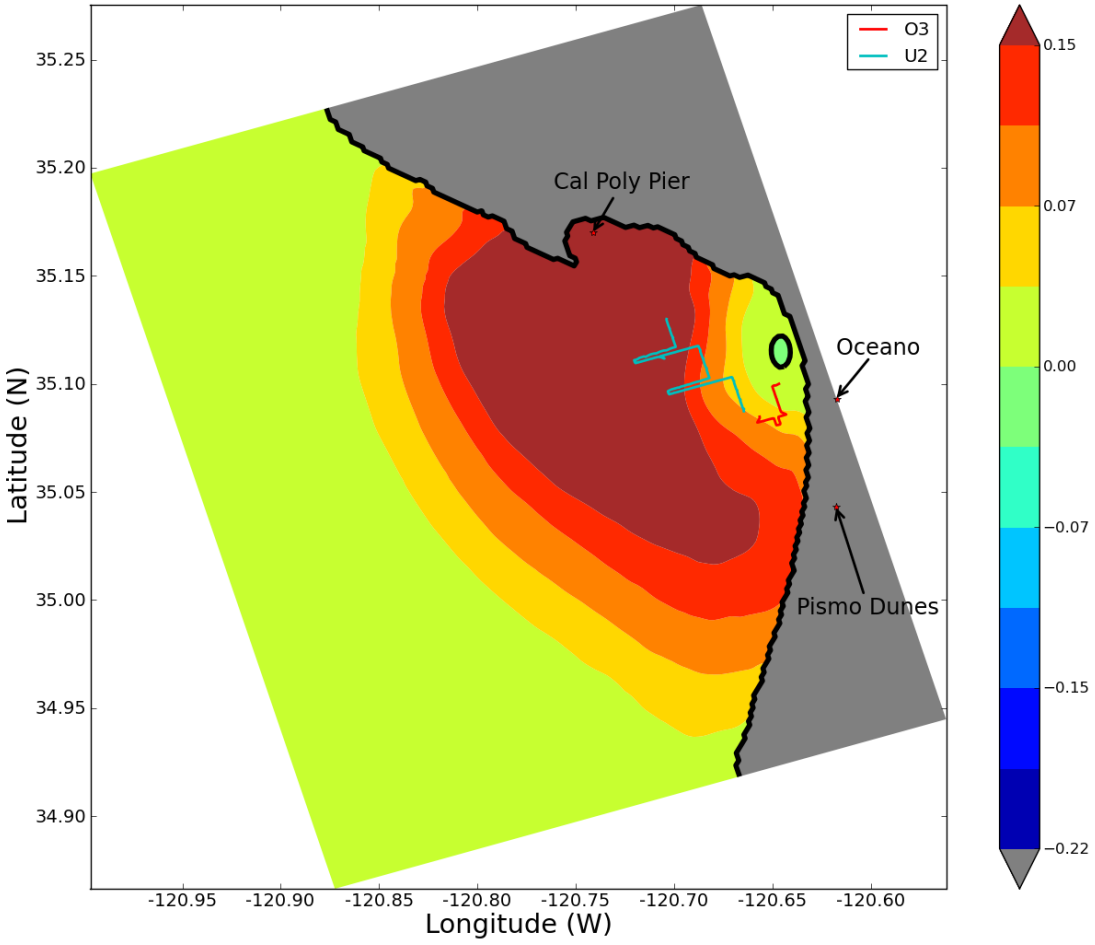

Figure 8.12: Standard deviation difference model: O3 - U2, original data

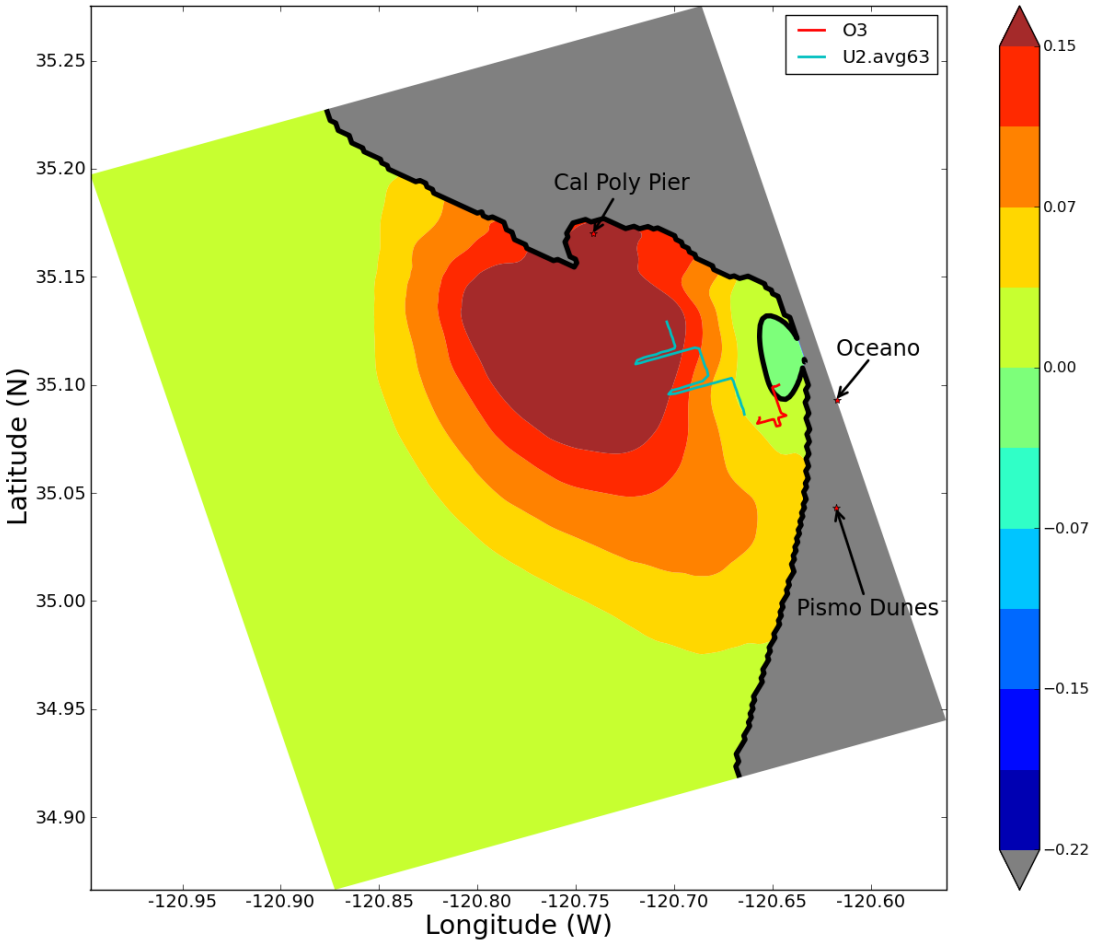

Figure 8.13: Standard deviation difference model: O3 - U2, points spread across range of U2 mission

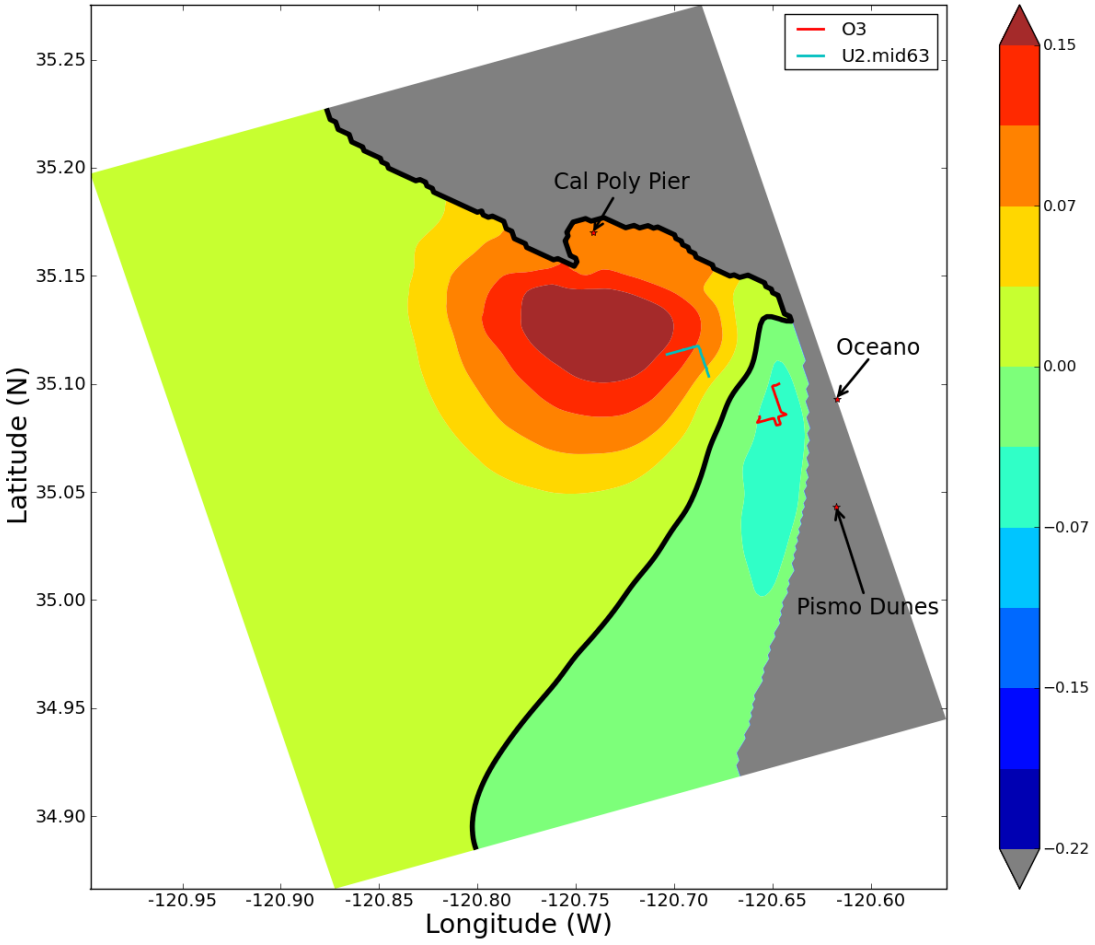

Figure 8.14: Standard deviation difference model: O3 - U2, 63 points in the middle of the U2 mission

Chapter 9

Analysis

The results of the experiment are many and provide for an interesting analysis. At a glance, the data does not appear entirely consistent. As one may expect, the optimized missions from the pier decreased the standard deviation of the assimilated model more than their unoptimized counterparts. However, the unoptimized U2 boat mission did better than the optimized O3 boat mission, no matter how the results (three interpretations) were compared. This section takes a closer look at the results.

9.1 Analysis of the pier missions

Both optimized pier missions reduced the standard deviation of the assimilated San Luis Obispo Bay model significantly more than their unoptimized counterparts. In comparing the shortest of optimized and unoptimized missions, O1 reduces the standard deviation 19.17%, 2.35% more than U1. In the longer missions, data acquired O2 reduces the deviation 19.79%, 3.38% more than U3. Even comparing the two missions whose distances are most similar - U3 and O1,

with 11948 meters of data and 12488 meters, respectively - the optimized mission is 2.76% better than the unoptimized mission.

It is interesting to note that even though the U3 mission is 3606 meters, or 43% longer than the U1 mission, it actually reduced the standard deviation of the original model *less* than U1. This suggests that the U1 mission visited points that in the original model were of higher standard deviation and therefore "more valuable" to the assimilated ROMS model than the longer mission.

The sums of the standard deviations from the original model also provide some interesting results (column $J(X)$ from Table 8.1). The sum of the 200 data points along U1's path from the original model (165) is actually *higher* than O1 (164). It was hypothesized in the problem statement that data gathered from points of higher standard deviation in the original model would reduce more effectively the standard deviation in the assimilated model. O1, though, reduced the deviation in the resulting model by 2.35% more than U1. This fact strongly suggests that it is more than just standard deviation that determines the output of a model.

9.2 Analysis of the boat missions

It is important to remember that O3 completed less than 10% of its mission, so the analyses offered in this section should be read with that in mind.

In this test, the unoptimized mission performed better than the optimized mission. Data from the U2 mission improved the resulting ROMS model significantly more than the O3 data - it was 8.45% better. It is necessary to note, however, that the distance and number of points used in calculating the model

was significantly larger for U2 than O3. There were 276 points over the 16293 meters for U2, while O3 only had 63 points over 4439 meters; this comparison could probably be considered *not fair*.

To make the comparison even in terms of data points, the U2 data was averaged to 63 points across the length of the data. While O3 did better than in the above comparison, it was still 5.83% worse than U2.avg63.

The data from U2 was adjusted once more so that it had both the same number of points and a similar distance to the O3 data (U2.mid63). The U2.mid63 data once again reduced the deviation in the resulting model more than O3. This set of U2 data reduced the deviation 2.39% more than O3.

As in the third paragraph of Section 9.1 above, the sums of temperature standard deviation from the original model were inverted from the sums of temperature standard deviation in the resulting models. In U2.mid63, the sum of standard deviation along the 63 points of the original model ($J(X)$) were 53, while for O3, they were 58. From the hypothesis, O3 should have outperformed U2.mid63 because it was significantly higher. Data showed, of course, that U2.mid63 was 2.39% better when comparing the assimilated models. This again suggests that more than the original model's standard deviation should be taken into account when developing missions.

9.3 Overall analysis

Though the results may not be consistent, one thing is very clear: the acquisition of any data helps to reduce the standard deviation, or uncertainty, of an assimilated San Luis Obispo Bay ROMS model. This is desired and expected,

but it is not the only conclusion that can be inferred from the data.

9.3.1 Diminishing returns

As Table 8.1 notes, even the shortest and "least effective" mission (O3) reduced the standard deviation more than 10%. At the other end, the "most effective" mission (O2; optimized) reduced the deviation nearly 20%. The longest and second "most effective" mission (U2) reduced deviation 19.61%. These facts suggest that "more (data) is better", but this relationship is not linear - there is to some extent *diminishing returns* with regards to the ratio of the sum of temperature standard deviation from the original model ($J_L(\chi)_{orig}$) to standard deviation reduction in the resulting model ($J_L(\chi)_{final}$), as shown in Figure 8.9.

The data acquired from O3 only covered 4439 meters while O2 data covered 13276 meters - 3 times more distance. O3's model was 11.16% better than the original model, while O2 was 19.79% better. At three times the distance coverage of the O3 model, O2 only improved the model deviation by 1.77 times over O3. This fact suggests that the distance travelled-to-deviation reduction ratio would look asymptotic: to get a 50% improvement from the original model, data may need to be gathered over a range of maybe 60 or 70 kilometers or more - more than *five* times the distance of the O2 data that was 20% better than the original model.

The scatter plot in Figure 8.9 supports this analysis. The best-fit line shows a small positive slope, which indicates that the vertical data (% reduction) is rising more slowly than the sum of standard deviations.

9.3.2 Not all data are created equal

In two of the three comparisons, data gathered from optimized missions produced a better model - to the tune of about 3% - than their unoptimized counterparts. These optimized missions were created from an algorithm that sought the best path possible given a distance constraint. The metric of "best path" was decided by that path whose sum of standard deviation from points in the original model ($J(X)$) was the highest. Figures 7.1 and 7.2 show clearly that the optimized missions spent more time in or towards areas of higher standard deviation.

Of course, the above assertion does not appear to hold true all the time - the model from O3 performed worse than U2 data, no matter how the U2 data was cut. Two of the comparisons (*U2 vs O3* and *U2.avg63 vs O3*; Figures 8.12 and 8.13 respectively) might be easily explained away based on the sheer difference of the range over which the data was gathered - more than 3 times. However, the last comparison - U2.mid63 vs O3; Figure 8.14 - turns the above assertion on its head. The same number of data points (63) were used across a similar range to create each model, and the path in the optimized mission was significantly "better" in terms of the sum of standard deviation from the original model.

There are a few explanations for this. The first could simply be that the first assertion and hypothesis - that data acquired from missions whose paths cover areas of higher standard deviation produce a better resulting model - is not always true. Another could be that data acquired close to shore - as it is in O3 - could have a shorter "half-life" than data farther from shore because of the complex dynamics of near-shore systems. Water moves more quickly near shore, waves kick up sediment, etc., and so it may be reasonable to suspect that data

collected near shore has a shorter expiration date as far as it affects the model. Yet another explanation could be the date or time of the data acquisition. The mathematical models in ROMS move winds, currents, etc. in the model according to output of deterministic geophysical equations; the ocean (T, S, u, v) may have been in a significantly different state on O3 than then were on U2.

9.3.3 System design

The system illustrated in Figure 5.1 above was successfully built so as to allow for the experiments and analysis discussed in this thesis. ROMS provided a model which was used by the path planner to develop missions for the Iver2. The Iver2 (in all but one case) completed these missions and provided temperature data to ROMS' data assimilation component. The data assimilation process produced a posterior estimate of the ocean model's standard deviation for each mission.

Every component of the system worked as it was intended. However, the results and analysis suggest that the system may need to take into account more than temperature deviation in mission design. This of course would require changes in output from ROMS, additional functionality in the path planner, additional sensors in the Iver2, and capacity for new information (e.g., salinity) in data assimilation.

Chapter 10

Conclusions

This thesis sought to develop a system through which one could explore the benefits of using a path planner over unoptimized missions for temperature data acquisition in San Luis Obispo Bay. I developed a hybrid path planner that combined the techniques of grid reduction and breadth-first search to efficiently explore an otherwise unwieldly model. The path planner was validated by the output of an Iver2-readable mission file, and its usefulness was tested by sending the Iver2 out to gather temperature data along that mission path. For comparison, the Iver2 also gathered data along unoptimized mission paths that were created manually.

The research conducted in this thesis shows, at the very least, that even a small amount of data collection will positively influence a resulting ROMS model.

The experiments also showed that in two of three cases, optimized missions were more effective in reducing the temperature standard deviation, or uncertainty, of the original model. In the third case, the Iver2 mission failed to finish its optimized mission, and so was only able to collect data along less than 10%

of the original path. That being said, even when this mission was compared to three different iterations of its unoptimized partner, it was still less effective than the unoptimized mission.

Lastly, a ROMS-based data assimilation system that intelligently plans for and integrates AUV measurements with the goal of minimizing model standard deviation was successfully developed. This system was used to carry out the experiments in this thesis and facilitated analysis.

I believe that this thesis and included experiments would benefit greatly from future work. A greater understanding of ROMS, data assimilation, and ocean processes could aid in the improvement of the designed system and model output.

10.1 Future work

The work presented in this thesis, while relevant and useful, is only the tip of the iceberg. Research and advances in path planning, data assimilation, and areas of robotics could be exploited to provide a more thorough, more robust, and faster exploration of the San Luis Obispo Bay model.

10.1.1 Acquire more and different data

The easiest and most intuitive next step is to "gather more data". The Iver2 only gathered temperature data, at its farthest, 13 km south of the pier. Near its southwestern border, the original model (Figure 6.3) has a large area where the standard deviation is high. In fact, the standard deviation in that area is higher than any other area in the model.

ROMS is capable of assimilating temperature, salinity, and water densities

and velocities. The integration of one or more of these data would certainly improve the San Luis Obispo Bay model. The Iver2, whose design is such that sensors and other components can easily be added or removed, could quickly be fitted with such a sensor and deployed as in the experiments. The missions could even remain the same.

10.1.2 ROMS research

While the above section may be the quickest and easiest next step, the path planning and acquired data could be most greatly improved by a better understanding of the modeling system that it uses.

The optimized missions were developed by using a year-old, static model whose data was only constrained by sparse wind data. Though the results were reasonable as far as where the error was reduced (i.e., in general, around where the data was taken), they were apparently contradictory as far as magnitude of standard deviation reduction when comparing the boat and pier missions (the U2 data outperformed the O3 data in all tests).

ROMS is a very complicated collection of software that uses discretized models of also-complicated, continuous geophysical equations (fluid dynamics, etc.). However, it is a software system, so it is implicitly deterministic, even if the subject of the model is stochastic. As such, there should be a closer look at why the boat missions did not perform like the pier missions. Depending on the results of *that* research, changes to the path planner might be necessary so that it takes into consideration the effects of the intricacies of ROMS.

Observation impact and sensitivity

One of ROMS many features is that it can provide an analysis of *observation impact* and *observation sensitivity* [21]. Observation impact refers to how much each data point contributed to the difference between the original model and the assimilated one. Observation sensitivity estimates the changes in the assimilated model as a result of changes in data or observations.

The path planning algorithm presented in this thesis assumed a binary observation impact - e.g., data gathered at x would reduce the deviation at x - without any awareness of the sensitivity. As the results show, the observation impact is in fact not binary - and it is not perfectly clear what the relationship is between the data and the resulting model. A more in-depth look and understanding of the observation impact and sensitivity may offer the greatest benefit to future models.

10.1.3 Real-time data assimilation and path planning

The path planning algorithm developed for this thesis was developed as a real-time algorithm that could be placed on the Iver2. Given a static model and end location, the Iver2 could quickly identify the best route to that location and start its mission.

The results of the experiments show that any amount of data assimilated into the original ROMS model has a positive effect on the output. It is proposed, then, that data acquired during the mission could be assimilated in real-time to produce another model mid-mission. This model could be significantly different than the original model; the best path through that model may be also be different than

the one identified with the original model. Real-time data assimilation is not currently feasible on the Iver2 (or nearly any commercial AUV, for that matter); this section proposes some solutions to that problem.

Rapid data assimilation using graphics cards

There has been recent research at Cal Poly exploiting NVidia graphics cards to achieve high levels of parallelization. There have also been discussions about mounting a graphics card in the Iver2 to take advantage of this parallelization so that on-board data assimilation could happen. If assimilated in or close to real-time, the resulting new model could be fed into an existing path planning algorithm (such as the one in this thesis) to update the "best" path.

Centralized data assimilation and path planning (off-robot)

Another option for real-time data assimilation is the offloading of mission data to a more powerful computing system (such as that used to create the models seen in the results). The assimilated model could be uploaded to the Iver2, which could use a path planning algorithm to update its path. Alternatively, the centralized computing system could also do the path planning for the Iver2, uploading only a set a waypoints to which the AUV would navigate.

10.1.4 Multiple robots

The experiments undertaken in this thesis could be greatly improved by acquiring data with multiple robots. A significantly greater area could be explored, as well as a larger number of data points obtained if more than one robot was used to acquire data.

Research in multi-robot systems, especially terrestrial, is a large field. Many of the algorithms and techniques used to coordinate multi-robot systems are agnostic to the medium in which it works. That is, these techniques can easily be ported to AUVs. One paper ([12]) discusses robot altruism to optimize task fulfillment. Another, specific to AUVs, discusses cooperative multi-AUV control as it was used in the Autonomous Ocean Sampling Network discussed in Section 3.1.2 [15]. There is no lack of research in the area of multi-robot systems; experiments like the ones undertaken in this thesis could benefit greatly from such systems.

Bibliography

[1] Airmar Technology Corporation. Airmar T80 Datasheet. http://www.blueheronmarine.com/files/Airmar_T80_Datasheet.pdf.

[2] A. Alvarez, A. Caiti, and R. Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *Oceanic Engineering, IEEE Journal of*, 29(2):418 – 429, apr. 2004.

[3] Autonomous Undersea Vehicle Applications Center. Search for AUVs by battery. http://auvac.org/resources/search/batteries.php, 2010.

[4] Autonomous Undersea Vehicle Applications Center. Search for AUVs by communication system. http://auvac.org/resources/search/comms.php, 2010.

[5] Autonomous Undersea Vehicle Applications Center. Search for AUVs by Navigation System. http://auvac.org/resources/search/navigation.php, 2010.

[6] Autonomous Undersea Vehicle Applications Center. Search for AUVs by physical capabilities. http://auvac.org/resources/search/capabilities.php, 2010.

[7] Autonomous Undersea Vehicle Applications Center. Search for AUVs by purpose. http://auvac.org/resources/search/purposes.php, 2010.

[8] Autonomous Undersea Vehicle Applications Center. Search for AUVs by sensor. http://auvac.org/resources/search/sensors.php, 2010.

[9] S. Behnke. Local multiresolution path planning. In *In Proceedings of 7th RoboCup International Symposium*, pages 332–343. Springer, 2003.

[10] T. Bishop, P. Tuddenham, M. Ryan, and D. Payne. Then and now: The HMS Challenger expedition and the "Mountains in the Sea" expedition. http://oceanexplorer.noaa.gov/explorations/03mountains/background/challenger/challenger.html, 2009.

[11] Center for Earth Systems Research. ROMS at UCLA. http://www.atmos.ucla.edu/cesr/ROMS_page.html, 2010.

[12] C. Clark, R. Morton, and G. Bekey. Altruistic relationships for optimizing task fulfillment in robot communities. *Distributed Autonomous Robot Systems*, 2008.

[13] R. Davis, J. Bellingham, P. Chandler, F. Chavez, N. Leoanard, and A. Robinson. AOSN II System Goals and Performance Metrics. http://www.mbari.org/aosn/AOSN_NewSite/Documents/AOSNII%20System%20Goals%20and%20Performance%20Metrics.pdf, 2003.

[14] R. Davis, N. Leonard, and D. Fratantoni. Routing strategies for underwater gliders. *Deep-Sea Research II*, 56(2):173 – 187, 2009.

[15] E. Fiorelli, N. Leonard, P. Bhatta, D. Paley, R. Bachmayer, and D. Fratantoni. Multi-AUV control and adaptive sampling in Monterey Bay. In *Au-*

tonomous Underwater Vehicles, 2004 IEEE/OES, pages 134 – 147, june 2004.

[16] S. Frolov, A. Baptista, and M. Wilkin. Optimizing fixed observational assets in a coastal observatory. *Continental Shelf Research*, 28(19):2644 – 2658, 2008.

[17] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *Robotics and Automation, IEEE Transactions on*, 8(1):23–32, 1992.

[18] J. Gould, et al. Argo profiling floats bring new era of in situ ocean observations. *Eos*, 85(19):179, 190–91, May 2004.

[19] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[20] A. M. Moore, H. G. Arango, G. Broquet, B. Powell, C. A. Edwards, M. Veneziani, D. Foley, J. Doyle, D. Costa, and P. Robinson. The Regional Ocean Modeling System (ROMS) 4-dimensional variational data assimilation systems, Part II: Performance and Applications to the California Current System. *Progress in Oceanography*, submitted.

[21] A. M. Moore, H. G. Arango, G. Broquet, B. Powell, C. A. Edwards, M. Veneziani, D. Foley, J. Doyle, D. Costa, and P. Robinson. The Regional Ocean Modeling System (ROMS) 4-dimensional variational data assimilation systems, Part III: Observation impact and observation sensitivity in the California Current System. *Progress in Oceanography*, submitted.

[22] Ocean-Modeling.org. Introduction to ocean modeling. http://www.ocean-modeling.org/index.php, 2007.

[23] C. Petres, Y. Pailhas, P. Patron, Y. Petillot, J. Evans, and D. Lane. Path

planning for autonomous underwater vehicles. *Robotics, IEEE Transactions on*, 23(2):331 –341, apr. 2007.

[24] Rutgers University Coastal Ocean Observation Lab. Autonomous underwater vehicles (AUVs) a.k.a. gliders. http://marine.rutgers.edu/mrs/projects/oceanrobots.htm.

[25] M. Soucy and P. Payeur. Robot path planning with multiresolution probabilistic representations: A comparative study. In *Electrical and Computer Engineering, 2004. Canadian Conference on*, volume 2, pages 1127 – 1130 Vol.2, May 2004.

[26] A. Stentz. The Focussed D* Algorithm for Real-Time Replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, August 1995.

[27] University of California, San Diego. HMS Challenger. http://aquarium.ucsd.edu/Education/Learning_Resources/Challenger/.

[28] B. Wang, X. Zou, and J. Zhu. Data assimilation and its applications. *Proceedings of the National Academy of Sciences of the United States of America*, 97(21):11143–11144, 2000.

[29] D. Wang. *Autonomous Underwater Vehicle (AUV) Path Planning and Adaptive On-board Routing for Adaptive Rapid Environmental Assessment*. PhD thesis, MIT, 2007.

[30] Wikipedia. Acoustic doppler current profiler. http://en.wikipedia.org/wiki/Acoustic_Doppler_Current_Profiler.