8-2021

Variable autonomy assignment algorithms for human-robot interactions.

Christopher Kevin Robinson
University of Louisville

VARIABLE AUTONOMY ASSIGNMENT ALGORITHMS FOR HUMAN-ROBOT INTERACTIONS

By

Christopher Kevin Robinson
M.S., Electrical Engineering,
University of Louisville, Louisville, KY

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Electrical Engineering

Department of Electrical and Computer Engineering
University of Louisville
Louisville, Kentucky

August 2021

VARIABLE AUTONOMY ASSIGNMENT ALGORITHMS FOR HUMAN-ROBOT INTERACTIONS

By

Christopher Kevin Robinson
M.S., Electrical Engineering,
University of Louisville, Louisville, KY

A Dissertation Approved On

July 19, 2019

by the following Dissertation Committee:

Prof. Dan O. Popa, Ph.D., Dissertation Director

Prof. Tamer Inanc, Ph.D.

Prof. Olfa Nasraoui, Ph.D.

Prof. Karla Welch, Ph.D.

DEDICATION

To my wife Jessica, mijn broer Joost, and my dear friends Joshua and Ellen Lancaster, without whom none of this would be.

ACKNOWLEDGEMENTS

I would like to extend my personal thanks to Dr. Dan Popa for bringing me to his lab and allowing me the exceptional level of education I've received working at NGS and LARRI, both for personal and academic growth. I would also like to express my gratitude to Dr. Tamer Inanc, Dr. Olfa Nasraoui and Dr. Karla Welch for their input, feedback, and time in being on my committee as well as contributing enormously to my education as a whole through the courses I have had with them.

To the University of Louisville Conn Center, I would like to extend my thanks for providing me with support through the early phases of my doctoral program when my status was uncertain, and also to the Logistics and Distribution Institute for further support during the later stages as well. Further, I would like to acknowledge the ongoing support of the NSF, through the KAMPERS and Future of Work projects ([1] and [2]) which has allowed my continued efforts toward application of the work herein.

I also could not have have gotten to this point without the support of my colleagues Sumit Das, Abubakar Shamsudeen, and Indika Wijayasinghe. I can't properly express my thanks for the amount of learning made possibly by the number of reviews, meetings, discussions and phone calls, nor can I for the extraordinary value of years of friendship.

I would also like to express my thanks to my family- my mother and father for supporting me and teaching me so many lessons I needed to make this happen, and for always being there to help me through the roughest times. And to my sister, for her wonderful, boundlessly enthusiastic support and dozens of rides to and from the university. En ook mijn broer van over de oceaan, Joost, wiens emotionele en intellectuele steun me op de been hield terwijl ik het anders niet had kunnen redden.

And finally, I am deeply and immeasurably grateful for the boundless love and sup-

port of my wife Jessica, who has stood by my side through such extraordinary circumstances on this journey that make me both humble and awed at the intensity of her love and strength. It is not hyperbolic to say that she deserves just as much credit for this accomplishment as I, if not more.

ABSTRACT

VARIABLE AUTONOMY ASSIGNMENT ALGORITHMS FOR HUMAN-ROBOT INTERACTIONS

Christopher Kevin Robinson

July 22, 2021

As robotic agents become increasingly present in human environments, task completion rates during human-robot interaction has grown into an increasingly important topic of research. Safe collaborative robots executing tasks under human supervision often augment their perception and planning capabilities through traded or shared control schemes. However, such systems are often proscribed only at the most abstract level, with the meticulous details of implementation left to the designer's prerogative. Without a rigorous structure for implementing controls, the work of design is frequently left to ad hoc mechanism with only bespoke guarantees of systematic efficacy, if any such proof is forthcoming at all.

Herein, I present two quantitatively defined models for implementing sliding-scale variable autonomy, in which levels of autonomy are determined by the relative efficacy of autonomous subroutines. I experimentally test the resulting Variable Autonomy Planning (VAP) algorithm and against a traditional traded control scheme in a pick-and-place task, and apply the Variable Autonomy Tasking algorithm to the implementation of a robot performing a complex sanitation task in real-world environs.

Results show that prioritizing autonomy levels with higher success rates, as encoded into VAP, allows users to effectively and intuitively select optimal autonomy levels for efficient task completion. Further, the Pareto optimal design structure of the VAP+ algorithm

allows for significant performance improvements to be made through intervention planning based on systematic input determining failure probabilities through sensorized measurements.

This thesis describes the design, analysis, and implementation of these two algorithms, with a particular focus on the VAP+ algorithm. The core conceit is that they are methods for rigorously defining locally optimal plans for traded control being shared between a human and one or more autonomous processes. It is derived from an earlier algorithmic model, the VAP algorithm, developed to address the issue of rigorous, repeatable assignment of autonomy levels based on system data which provides guarantees on basis of the failure-rate sorting of paired autonomous and manual subtask achievement systems.

Using only probability ranking to define levels of autonomy, the VAP algorithm is able to sort modules into optimizable ordered sets, but is limited to only solving sequential task assignments. By constructing a joint cost metric for the entire plan, and by implementing a back-to-front calculation scheme for this metric, it is possible for the VAP+ algorithm to generate optimal planning solutions which minimize the expected cost, as amortized over time, funds, accuracy, or any metric combination thereof. The algorithm is additionally very efficient, and able to perform on-line assessments of environmental changes to the conditional probabilities associated with plan choices, should a suitable model for determining these probabilities be present. This system, as a paired set of two algorithms and a design augmentation, form the VAP+ algorithm in full.

In summary, this work presents the following three material contributions:

- Formulation of the VAP algorithm, intended for the study of human reasoning in the context of autonomy level selection
- Formulation of the VAP+ algorithm, implementing principles identified via the VAP algorithm to autonomous autonomy level assignment
- Application of both the VAP and VAP+ algorithms to the practical problems of Robotic Bin Picking Tasks and Robotic Surface Disinfection

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this chapter I will be discussing prior research which relates to both the extant work in the area of variable autonomy, as well as the development of concepts which underwrite the necessary concepts at play, and finally discussing the contributions of this research. As such, this chapter seeks to fill two roles: firstly, to set the stage for my contributions by illustrating the state of affairs in the field, and further to present the necessary supporting material for my key arguments in favor of the benefits and motivation for my work.

1.1 Background

As autonomous robots become increasingly more sophisticated, applications for use in environments beyond the traditional industrial material-handling niche have opened up. Of particular interest to this investigation are automated assistant-type robots which are capable of operating in human environments while performing tasks which are difficult or unpleasant for the operator to accomplish.

Robotic systems that share immediate spaces with humans in living environments pose a unique set of safety challenges because of the complexity of unstructured environments [3], but recent advances in collaborative robotics and advanced perception algorithms has driven the adoption of robotic systems into new venues. This expansion presents a prime opportunity to alleviate difficulties for humans on jobs that can be augmented by robotic tools. Autonomous systems can reduce cognitive loading from repetitive or focus-demanding tasks [4]. Furthermore, there are many cases, such as hazardous materials handling, for which it is impractical or unwise for humans to perform the task in question [5].

Despite the benefits robots can offer, there are still significant gaps in their ca-

pabilities. Many researchers have proposed to supplement robot autonomy with human perception and decision making, effecting a traded-control scheme [6]. Given that there are advantages to certain parts of each participant's repertoire, is it natural to divide the task load to humans and robots in accordance with their optimal capabilities. On the other hand, when the operator and robot both contribute to a teleoperation task achievement concurrently, the operational paradigm is defined as shared control [7]. The designation of which subtasks, and what proportion of total work should be assigned to human or robot refers to the level of autonomy of the system [8], a topic that has been increasingly studied with the advent of autonomous vehicles, the DARPA Robotics Challenge, and so on.

Though the full topic of shared control teleoperation carries an implicit dependence on partitioning task spaces into human and machine components, in this paper we focus on the specific domain of variable autonomy (often 'adjustable autonomy' when an operator controls the level of autonomy) meaning systems for which the degree of control the operator asserts is not assumed to be fixed throughout the task performance. Variable Autonomy has an established history as an effective response to the inherent gaps in robotic functionality. In [9] and [10], research pertaining to human intervention is presented with the example case of bin-picking which resolved errors by elevating control to a human operator in the case that a fault is detected. These examples employ only after-the-fact error detection, a single autonomy level, and primarily focus on intervention by way of fault analysis.

The work in [4] presents a study on the quality of interaction between human users and an autonomous vehicle related to the level of autonomous action present in the vehicle. The paper presents user studies to assess the impact of different levels of autonomy to the quality of autonomous driving. An explicit discussion of the impact of autonomy level in the context of human-robot cooperative tasks is given in [11], in which the authors describe a variable autonomy system based on the concept of action and neglect. In the absence of teleoperated input from the operator, an autonomous system function was triggered. The autonomy switching trigger was explored in the context of delay between human input and robot, rather than environmental observation. Additionally, work in [12] presents a

novel interface for a human to a robot using an input device which incorporates variable autonomy into its control structure. This paper focuses on aspects of the device as an input system, and in that context approaches the topic of differing levels of autonomy, but without expressly assigning the user control over the autonomy level, rather working to address the issues associated with changing input characteristics due to shifts in autonomy level.

In complex, changing environments, the range of tasks is sufficiently broad that a general-purpose robot would need a multitude of bespoke modules to adequately function. Furthermore, the rapidly changing workspace conditions may often render a given task model irrelevant in a momentary context and give rise to certain situations in which the robot's autonomous capabilities are more likely to degrade. It is also possible that over-reliance on operator interventions may unduly tax users, introducing inefficiencies to the system. It is therefore appropriate to adjust the ratio of human-driven tasks to robot tasks in accordance with the user and the current situation [13]. There are other similar examples, in which autonomy is positively correlated with speed and negatively correlated with accuracy, as in the case of human-computer interface performance expressed by Fitts' law [14]. Fitts' law establishes that there is an inverse relationship between the complexity of a task and the speed, presuming a constant 'information throughput', IP. In our case, IP can be likened to task completion success, and so we expect a similar conditional relationship between task complexity and the corresponding success threshold.

It has been established, for systems with challenged autonomous modes, that optimal performance of the human-robot system occurs at an intermediate level of autonomy [15][16]. Additionally, research has shown that users are quite capable of determining appropriate levels of autonomy [17]. However, common definitions of autonomy are such that implementations utilize ad-hoc assignments of robotic task modules to specific autonomy levels. For instance, a rigorous discussion of identifying autonomy levels can be found in [18], absent a metric for assignment of actions to specific levels.

1.1.1 Importance to HRI

In the context of autonomy as a parameter, human/robot interfaces very naturally become important quickly. In any case in which autonomy is not fixed and of a total nature with regards to functionality, there is a point at which human interaction with the robot becomes critical [19]. While it is common to assume that HRI refers first and foremost to direct physical interaction between a human and a robot [20], it is also common to express an interaction through the medium of a technical interface [21]. There are very nearly as many interfaces between humans and robots as there are robots: gesture control systems [12][22], human monitoring safety systems [23][24], brain-controlled interfaces [25][26], mobile device-based interfaces [27][28], base-station controllers [29][30], voice operated controls [31], and a plethora of other means. One common feature of interfaces is that they, in some way and to some measure of authority, allow the human to control the operation of the robot.

Human control of a robot is of course not the only means that an interface can provide, often times an interface allows a robot to present information or make requests of a human [32], but in most cases a robot is in some way influenced by the human input [33]. From this perspective, that an interface is typically a way for an autonomous system to be directed by a human, it becomes naturally clear the degree to which autonomy is influenced by the interface, and in particular the paradigm under which the interface presents agency to the human. Such paradigms may include direct, total control to the human such as in teleoperation [34][35], limited control means as in shared control [36] or traded control [37], or only the capacity to set directives [38][39].

Of particular note for our purposes is the fuzzy nature of traded and shared control schemes. Such models may include periods of time in which, for all intents an purposes, the robot is being teleoperated by the human [40][41], or cases in which the human has a relatively high degree of agency over the robot, but active control subsystems are necessary for operation [42][43][44]. One of the most pervasive examples of this latter case being fly-by-wire systems [45][46] in which manual management of all the systems required to effect control at the highest level is simply impossible for a human.

What unifies all these cases is that in every circumstance, the system can be said to be expressing a certain level of autonomy [47]. While an aircraft and a consumer automobile both have humans controlling input hardware to effect navigation, the explicit degree to which the human possesses low-level control is significantly different [48]. Further, the difference between a 3-DOF gantry system controlled by linear actuators and a 6-DOF robotic arm constrained to 3-DOF by a joystick also exhibit vastly different levels of necessary autonomous integration. The central point here being that HRI systems are, by definition, invested in the definition of levels of autonomy because they operate at an assumed level beyond full-autonomy.

Because they are inherently operating at scaled autonomy, they are also inherently subject to performance constraints inherited from the comparable effects of variant competencies between humans and robots. This means that any system which can respond dynamically at any level will benefit from a rigorous definition of level of autonomy, by virtue of impacts of optimization repeatability on, minimally, interface design.

1.2 Traded & Shared Control

In most systems, the common application case for integration of a human to a robot is traded control [29]. In particular, when the shallow distinctions separating traded control and shared control are relaxed [49], then the range of systems classified as such increases greatly [4]. The common concept behind traded and shared control schemes is that the autonomous system relies on some degree of human input. For traded control, this means that the operator periodically assumes direct control authority over the robot [50], such as when a mobile robot is driven by control of it's motion vector. By slight contrast, shared control is when an operator and a robot possess relatively comparable influence [7] over the actions of the system, such as a mobile robot which effects trajectory control autonomously on the execution of a human-chosen navigational path [51].

Looking at these two cases side by side, however, it is clear how the distinction is more a spectrum than clear category: in the former, the robot likely implements some

control system to effect vector directives as actuator motions [22]. In the latter, while the human sets the path only, they likely heavily influence the action of the trajectory execution system through stylistic changes in the paths [51]. It is plain to see that the labels of 'traded' and 'shared' control most properly refer to regions along an autonomy spectrum in which the balance of agency shifts [16].

1.2.1 Variable Autonomy

Research has established that for systems with variable autonomous levels, optimal performance typically occurs at an intermediate level of autonomy [52]. Comparable to the economic concept of 'comparative advantage', when robots and humans have asymmetric competencies, even if one agent has an absolute performance advantage over the other performance gains are possible when dividing total labor [53].

Additionally, research has shown that users are quite capable of determining appropriate levels of autonomy [54]. However, common implementations utilize ad-hoc assignments of subtasks to autonomy levels. A rigorous discussion of identifying autonomy levels can be found in [55], however lacks a metric based approach for their assignment. An interesting case exists in [56], in which the authors utilize Markov decision processes, but focused on Human-in-the-loop controls systems.

Variable Autonomy has been used to supplement to the inherent gaps in robotic functionality. Such as in [9] and [57], where elevation of control to a human operator when a fault is detected is examined in the example case of resolving errors in bin-picking. These examples employ only two autonomy levels, however. Sliding-scale autonomy has also been described [58] as a means to develop robustness with respect to operation of robots in challenging environments. Furthermore, [59] discusses ways in which multiple autonomy levels can affect a response to difficult environmental conditions in control of a teleoperated robot. [60] presents a method using temporal logic to construct Pareto-Optimal policies, specifically for the implementation of switching functions for a switched mode controller.

1.2.2 Mixed-initiative Autonomy

A more recent approach increase autonomous system involvement in the selection of autonomy level is Mixed Initiative (MI) Human-Robot Interaction, which is defined in [6] as a system in which humans and robots both possess authority to intervene in task achievement. [61] presents a design methodology for constructing MI controllers based on expert evaluation of the tasks being performed, and mode-switching fuzzy control system for managing changes in level of autonomy. [62], by contrast, presents a collaborative architecture which can react or preempt changes in autonomy level on basis of high-level mission objectives. [63] presents a fully proactive implementation by incorporating concepts from Model Predictive Control using future predicted states of the system. [60], similarly, utilizes Markov decision processes as a system model, from which effort-based cost functions can be derived in relation to performance characteristics.

Mixed-initiative variable autonomy attempts to approach the problem, inherent in variable and sliding scale autonomy cases, of assignment by enabling the operator or robot to seize control when performance metrics or judgement indicate low performance of either. These MI systems are, in practice, frameworks for design of individual systems. They present methodologies of integrating performance metrics into planning and control system levels which ultimately rely on design decisions made at time of system implementation.

1.2.3 Machine Learning

An alternative solution to this problem is via the application of machine learning. Machine learning can be used to learn an optimal policy for selecting autonomy levels based on data available to the system. [64] demonstrates autonomy level switching controlled by performance prediction through a reinforcement learning system. [65] uses the MAXQ algorithm to learn optimal policy choice under the assumption of a Markov decision process, where the subprocesses are individual tasks, with 'human control' being a possible subtask which is consistently available. Both of these use variants of reinforcement learning, whereas [66] develops a neural network which learns to set the optimal level of autonomy on basis

of optimizing time-based costs.

While varied, existing methods of applying machine learning to variable autonomy all implement optimization on problem-specific objective functions, and further suffer from the same limitations of machine learning in general, namely training-specific policy identification and the requirement for ample and representative training data to be available.

1.2.4 Intervention

Given this perspective, another way of looking at human influence on the autonomous agent action which is, in context of the goals of our work, more readily applicable, is the concept of intervention [67]. Intervention models presume that, throughout the action of the autonomous agent, there will be times in which the robot needs human assistance [68]. This action may take the form of manual operations spanning a wide range of interfacing which, taken alone, span the spectrum from top-level directive setting all the way to teleoperation, or even independent actions of a user in support of the robot.

Going forward, the intervention model of HRI for task accomplishment will be the predominant model we will be using, with the presumption that subtask accomplishment will be the effort of dedicated design for the specific task. In particular, when considering the recursive decomposition of tasks down to minimal components which do not yield an intervention-based decomposition, we will presume that the full system we are investigating is the joint structure of this nested breakdown, and that all modules thus considered 'subtasks' are functionally atomic [69].

1.2.5 Problem Statement

The core problem to be addressed, then, is of determining when a human's capabilities should be deployed to effect actions of the autonomous system. The recognition here is that intervention bears costs [20], in terms of time [67], effort [70], probability of success [58], and many other potential metrics which influence overall performance [47]. Due to this, there will naturally be inherent factors which determine when and how human's and

machine's work is best put to use in task completion.

Pursuant to this observation is also the annotation that, tautologically, there must be a divergence in competencies between manual operators and robots [71][72]. Were this not the case, outcomes of intervention or non-intervention would be, for all intents and purposes, identical, and thus the problem would be to simply pick the lowest cost choice for any task. At the most broad level, this would render either human workers or automation irrelevant, but this is obviously not the case. Given that there are differences in competencies, then, it is a question of optimization to most effectively utilize these competencies [68].

When there are multiple options for achieving a goal, with divergent benefits and detriments, there is generally a cross-sectional optimum which can be achieved by examining the combinations of elements, under the constraints established by the task environment, and identifying the most efficacious such combination [73]. This presents a two-fold problem outside of theory, however, via the joint problems of dynamic environments and combinatorial explosion. The latter is the fundamental limit on tractability in evaluating all possible outcomes of an exponentially growing set. The former the complications due to the inherent variance in conditions, whether natural or artificial, which impact the evaluation of the combination system comprising the full state space.

Fundamentally, what these effect result in is the need for effective planning algorithms which, in one way or another, leverage problem structure to eliminate problems and develop robustness to environmental changes which may disrupt predictions [39][71]. The uses of objective optimization functions [74], classical planning [75], and machine learning [76] towards intervention planning have all been demonstrated on many application cases, however there is a fundamental weakness in the area of generalizability of these strategies [5].

There are two principle issues: Firstly, that many approaches are bespoke, using specific problem structure to develop a solution, for which problems not containing the same kernel processes are totally alien; second of compatibility between use cases, in which specific utility embedded within a more abstract method does not translate to an alternate

case because of absent means of adaptation, even if descriptors of the two problems are applicable [77].

The second issue is best viewed as a manifestation of unreliability of means for discussing autonomy. When a system is proposed, as many have been, for arranging and planning under a variable autonomy system which effectively manages the benefits and detriments of an autonomy scale change, but depends on a designer specifying the system's level of autonomy in an ad-hoc process, the chances that the designer selects an ill-conditioned set are significant [20]. Even more critically, any analysis of an intervention system which relies on the specified framework falls apart on the notice that, even if the framework offers some demonstration of efficacy, that a free human design element will undermine any and all postulates supporting the analysis.

This leads to an additional complaint which is that, as in much of behavior-based robotics, it becomes very difficult to provide operational guarantees without a very specific evaluation of a fully complete system. Further, in most cases any changes to the system will invalidate those predictions, necessitating either a complete evaluation of all possible situational variants, or very weak bounding on behavior [78]. In many cases, the only extant proving for a system under intervention is experimental evidence, excepting those systems which are principally constructed under the models of controls theory.

From this discussion, we can refine our core problem to two components: a lack of mathematically rigorous autonomy level assignment, and a lack of a corresponding means to perform planning on systems thus assigned. We will be addressing this pair of problems by presenting two algorithms, each of which addresses both problems under differing contextual assumptions.

Our problem is thus to construct a mathematically well defined, generalized mechanism for assigning tasks specifically to autonomy levels based solely on measurable data, and using a structurally independent system model. Our goal is the construction of such an assignment algorithm, without recourse to any problem-specific design criteria and in such a way as to allow us to give some measure of performance guarantees for the algorithm.

1.3 Motivation

In this section I discuss the ways in which the concepts described in the prior section interrelate and provide the rationale behind the VAP and VAP+ algorithms, as well as the value of pursuing them. I begin with a survey of differing definition work in Robotics, naturally essential for a thorough examination of Autonomy itself as a metric property of an autonomous system. With autonomy defined as a property, I proceed to work focused on defining differing levels of autonomy within a system- that which seeks to apply measurable scales to autonomous systems.

With efforts to establish measured autonomy levels defined for a system, I proceed to the parallel topic of assignment of autonomy level. It may seem that this would be better served embedded within the prior section, however it is important to recognize that the problem of specifying placement on an autonomy scale as an abstract task, and determining the most appropriate distribution of tasks across that scale, require significantly different methodologies.

With autonomy level assignment addressed, the next relevant topic is then research on systems in which the level of autonomy can be changed. Work on this topic includes both studies of the effectiveness of implementation approaches to variable autonomy levels, and efforts towards design of these systems. This review is critical because this work sets the specific context for my research, which specifically addresses outstanding complications which arise in sliding scale autonomy systems.

Finally, I discuss research in the arena of adaptive autonomy. This refers to systems in which autonomy level is, in some way, reactive to environmental pressures. Though the principle motivation for the development of the VAP and VAP+ algorithms is rooted in problems associated with variable autonomy, properties of the solutions I present can be leveraged to directly accommodate adaptive systems, which properties drastically improve the value of the algorithms as practical implementations for automated planning.

1.3.1 Autonomy in Robotics

There is a common definition of the word 'Autonomy', which incorporates tenants of independence, self-sufficiency, and agency. While this colloquial definition does encapsulate many of the properties that are essential to an understanding of autonomy in a robotic system, it does not present a level of rigor which is suitable for analysis in a scientific domain.

Fortunately, significant effort has been put forth in developing such a definition for use within robotics. Less fortunate, however, is that many definitions have been proposed, without a particularly strong consensus on which should be adopted in general. Often, in writing about autonomous robots, authors will either work within the auspices of the colloquial definition, or simply adopt one of the many options which is most fitting and convenient for their purposes.

In general, this situation does not often present much of a problem- in most cases the efforts are sufficiently qualitative in nature as to render specificity of the definition immaterial. However, for a more pointed purpose such as my aim to define and assign autonomy level rigorously it is necessary to adopt a definition which well-bounds the consequent level definitions. In this section, we discuss several such definitions, analyzing the merits and detriments of each under comparison.

1.3.2 Defining Levels of Autonomy

A direct consequence of most definitions of autonomy, excepting some of the very most restrictive, is the natural conclusion that autonomy is not a binary property, but rather a continuum. Certain simple systems may permit a firm distinction only between states of being autonomous or non-autonomous, but the threshold at which a panoply of states becomes available is very low.

For instance, one may define a 'robot' as being a system which contains all elements of the Sense/Plan/Act (SPA) paradigm. In this light, we may consider a common traffic light with simple interfaces a very primitive robot: Pressure plates in the road provide

input, timing rules the plan, and the lights themselves the action means. Further, many traffic lights allow pedestrians to influence traffic patterns through the call button. In this case, there are autonomous systems with a modicum of human control that is not absolute, representing a point between autonomous and manual control.

However, this begs the question 'How autonomous is this traffic light?'. Establishing the nature of autonomy leads directly to the conclusion that there is a metric parameter, but the issue of specifying a useful comparison between these distinct measures is difficult. In the example, one might get the intuitive impression that the level of autonomy is fairly high, as the pedestrian's control over the traffic light does not generally represent a significant correlation between effort and outcome.

There are, generally, two rough kinds of categories which can be applied to models for autonomy levels: operational definitions, in which autonomy level is expressed in terms of the effects of the levels on activity, and assertive definitions, in which autonomy level is defined based on the content of the levels themselves.

To illustrate this, we might consider the hypothetical operational definition of time allocated to control. Suppose that we elect to make autonomy level be measure as a function of the amount of time spent asserting control over the agent. In some cases, such as a basic navigation task, this is a simple measure because time and activity are isometric, but it can also become significantly more complicated. For instance, in a picking task a robot may easily spend in excess of 90% active time under robotic agency. However, there may be specific, pointed cases in which intervention from a human is necessary. If this is the case, is it really valid to suggest that the robot controller is 90% autonomous? For some applications this may be valid, but a broad definition of autonomy levels would then have to account for the situational variability.

An alternative definition might be assertive- in the same picking task, we might have eight different sub-tasks which, together, represent the system's entire workflow. We might then define autonomy levels based on which of these modules is human-driven, and which is machine driven. Perhaps in this case, two of the subprocesses require direct control by an

operator, and we therefor define the autonomy level as being 25% autonomous. It is fairly easy to make a similar argument to this approach as the last, as we might assign other relative impact factors which shift the validity of the module control paradigm.

Looking at it this way, we can see that there are many different proxies for autonomy we might use, whether asserted by the designer or empirically measured- time, effort, energy, control authority, correlation to effect, value of effort, cost, and of course, any combination thereof. One might even decide to consider autonomy an abstract property which must be measured experimentally for any given implementation case via analysis of many of these factors independently.

This diversity of options is reflected in research. Across the number of publications engaging with levels of autonomy, there are many, many methods proposed. These proposals include efforts from those that are focused on studying effects of autonomy level, those specifically defining levels, and those implicitly defining them though the creation of a system with a targeted level of autonomy. A common property, however, across all this variety is that the structures proposed tend to be highly qualitative, and typically support only an ad-hoc application to design.

There have been a number of publications specifically seeking to define autonomy levels in an all-encompassing way, broadly enough to encompass all fields of automation. These most resemble taxonomic classifications systems. Levels are defined based on observed action, and are typically qualitative in nature- hence the relation to taxonomic classification. Though providing a sufficiently broad framework to classify all or nearly-all robotic systems, they do not typically provide much in the way of design tools. Were one to ask 'what level of autonomy is my robot at?', then the answer would almost always be immediately apparent. However, if one were to attempt to align their design efforts around this- 'What level of autonomy should I program my robot for?', the classification system would then be of little value.

The real problem with the impacts of these classification systems is not in their utility for analysis- an experiment seeking the effect of autonomy level on trust, for instance,

is well served by these taxonomies for seeking correlations. Rather, the problem lies in the effect on the application phase. In most cases where autonomy level is railed to one side- full manual or full autonomous- the selection of components' responsibility between human and robot control are ad-hoc. A designer will, usually, pick which modules to automate based on externalities such as the difficulty of the problem and pragmatic design constraints. The autonomy level, therefor, might allow predictions about user trust, but design for improved trust becomes subordinate- one might attempt to design for trust, but the taxonomy provides no tools besides a goal.

1.3.3 Assigning Levels of Autonomy

Beyond the difficulties expressed in simply determining how to define the nature of an autonomy scale, there is the further complexity of attempting to assign any given systematic implementation to a level. As alluded to in the prior section, a means of leveraging the analytic power of a level classification towards design benefit is contingent on a developmentally oriented use of that classifier. To do so, it must be possible at the very least to take hypothetical delineations of task assignment between the human and the robot onto level classes.

For example, say a designer has multiple implementation models, each with a different level of autonomy. In our picking example, we might say that she is able to implement object recognition and grasping both as manual and automatic modules, an thus has four different combinations of manual and autonomous systems to consider.

Firstly, having both components manual would clearly be the least autonomous system, and having both automatic the most. We might deduce from this problem description that these two components are the only ones under consideration for manual operation, and therefor all other components are autonomous. Consequently, having both recognition and grasping autonomous would naturally correspond to the maximal autonomy version of the system. It is less clear into which category the version including two manual modes would fall, and the assignment would clearly be based on context of the rest of the system.

More critically, we then have the single manual phase options- under a pure system of counts of manual subtasks versus autonomous subtasks, we end up with an uncomparable set- two different configurations both correspond to the autonomy level defined by having one manual subtask. It is then easy to see how this inconclusivity may lead to problems in implementation. For any one problem it might be readily obvious which of the configurations to choose. Perhaps in our example (for any reason one could imaging), considering the grasp-selection module the 'level 1 autonomy' system is ideal. But the selection may not always be obvious, and at higher complexities even evaluating the possibilities may be intractable vis-a-vis combinatorial explosion. For instance, a system with 5 subtasks to choose between, the 'level 3 autonomy' case would include 125 different configurations to consider.

One might be tempted to suggest that simply picking among the many options at any level would be fair means to acquire design-based benefits associated with staged autonomy levels. Under that paradigm, the task would be to select autonomy level based on the desired benefits of each level, and then select from among the possibilities the specific version of that level implementation based on other design concerns. This would be a viable option, however the non-comparability of the grouped levels is itself a fundamental inversion of the conditions under which experiments seeking correlations between autonomy level and relevant design requirements fails. In metaphor, it is a compression of a non-linear state space to a linear one, with all the attendant risk of approximation error- it is not theoretically valid to trust the linear scale results when the relevant system is inherently nonlinear.

So, while the assignment of the hypothetical maximum single level of all manual options illustrates the fundamental scaling problem associated with mapping a task space onto an autonomy scale, the more pressing concern is the effect of using results related to a scale which is dimensionally inappropriate for the task at hand. Though one may observe the corresponding benefits from making the approximate choice, there is no guarantee that they will be consistent or predictive, and critically cannot be guaranteed to be sustained. For instance, in the picking task example, our designer my select the grasping module to be manual, and observe a corresponding user trust improvement correlated to 'level 1

autonomy' results in the literature, but she cannot assume these results will hold if she switches to the object recognition version of 'level 1 autonomy'. There is always the chance that she picked the 'correct' version of level 1 for the framework she is referring to.

Given this context, the need for reliable and rigorous assignment methods for allocating subtasks, in application, to autonomy levels is essential for the ongoing scientific validity, as expressed as predictive power, of autonomy level classification systems. Should a three-component system have a single module which itself represents (as far as level-dependent results are concerned) a higher autonomy investment than the other two components combined, that must be reflected in the assignment system. An assignment system represents a bridge between the categorical, quantitative measures associated with autonomy levels as exists in the literature and tangible properties of an arbitrary robotic system.

This is not to say such structures do not exist. There are in fact many such frameworks extant in literature which seek to perform this assignment, typically chosen so as to mesh with a well-known established classification system. Many of these frameworks are, as we will see, quite sophisticated. However, they suffer in general from a few specific weaknesses. By connection to inherently qualitative classifiers, they inherit many of the weaknesses related to ad hoc judgement, even when discretizing the scales. Most such methodologies also implement classification more or less by implementing a semantic decision tree, which again means that human judgement and biases are intrinsically built into the framework. Finally, and most importantly, by virtue of being based on fixed scales which are not connected to strictly measurable properties, they lack fundamental rigor and repeatability guarantees which can be provided only by mathematically-derived metrics.

In any circumstance in which autonomy can be divided into multiple categories, regardless of the means, there is the further decision to be made regarding the level of autonomy the system ought to express. In the prior sections, we have been examining properties of autonomy scales under an invariant paradigm in which the level is considered, essentially, a design parameter: it is selected during the testing phase and deployed as a constant. However, it is quite obvious that in many instances, the effectiveness of a given

level of autonomy is likely to not have a single global optimum. All engineers are intimately familiar with the conditional variance of system effectiveness as a function of environment, and while it is true that often engineering design is only able to effect the aggregate average performance this is usually not the case in intelligent systems. It is therefor ideal to consider that a system operating at the average best-case autonomy level may severely under-perform compared to a system which can adopt multiple levels to suit changing conditions.

The primary complexity occurs when the question of how to select what autonomy level to operate at any given time. Sometimes, this may be direct and simple, but often the correlation is predicated on variables not observable to a robotic system; one of the most urgent and pressing being the psychological state of a human- both critically impactful and notoriously difficult to measure. Given this difficulty, one of the most potent and elegant solutions has been to assign the task of autonomy level selection to the human.

Human intuition has, under suitable incentivization, consistently proved to be a viable means of coping with uncertainty in partially automated systems, and human intervention remains the primary recourse of eliminating problems which machines are ineffective or at least inefficient at solving. In essence, intervention planning is itself an optimization problem, in which we view the human and the robot as two separate subsystems with independent competencies and constraints, and attempt to best leverage the cross sectional optimum between both.

Through this lens, we can see that sliding scale autonomy presents a solution- selecting which set of competencies to use is a competency which human operators are far better suited for, in most cases, than their autonomous counterparts, and so presuming to assign that responsibility to them is a naturally optimizing option. This, however, puts a significant load on the design of each system to convert a problem to a sliding scale friendly mode. The autonomy scales and assignments described prior are all viable options for defining and allocating work to the scale function, but for a human operator to make good use of it, the scale must be sufficiently correlated to their perceptions.

This recognition brings us full circle with regards to the weaknesses in ad hoc

definition- if the designer's perspective is not aligned with the user, then the resultant sliding scale will be ineffective or possibly even deleterious to performance. Engineers and end users often have vastly different perspectives regarding machine interfaces, and so this is a high-impact consideration. Additionally, biases built into the autonomy classification framework and expressed in implementation become fundamental flaws when planning optimization is attempted based on data.

Given these limitations, the motivation for implementing scaled autonomy but under a context in which rigorous empirical definitions of autonomy level becomes immediately clear- the application of rigor to defining the scale used in the sliding scale is a means to eliminate the unpredictability and uncertainty implicit in a system which utilizes qualitative measures.

One of the natural results of converting a design-driven approach to sliding scale autonomy to an empirically driven approach is that it becomes far more possible to accurately and reliably implement adaptive techniques to the planning algorithms resulting from the systematic implementations. This is a direct emergent property of the measure based paradigm of autonomy assignment- if the assignments are based on measures, then reactive observation of the measure allows for adaptive control of the assignment.

Adaptivity is not, however, strictly contingent on fully rigorous definitions of autonomy, and in fact there is a large body of work devoted to adaptive autonomy. Often times, work implicitly expresses that adaptive features are implemented as a means to control undesired results of uncertainty- which we have seen is clearly influenced and magnified by adoption of scale systems which lack rigor.

Among the work observed in adaptive autonomy systems, it is notable that the means of altering performance in response to environmental variation are typically implemented as reactions to unacceptable performance of the system. That is to say, being reactive modifications, the common application of adaptivity seen in extant research is typically not due to an influence to optimize a system, but rather to eliminate undesirable behavior observed in work on which the adaptive system builds.

My point in this case is that while ad-hoc adaptivity designs do present a means to mitigate errors, an entire region of performance is evaded by not beginning with adaptive systems being the goal. Beginning the optimization process with a system capable of reacting to all observational data has the potential, especially when coupled with powerful machine learning techniques, to access operational domains invisible to the designers. Through structural adaptivity which is embedded in the automatic planning structure, and which can incorporate changes in systematic response without design effort, the flexibility of online learning can be immediately coupled to the planning algorithm.

1.4 Contributions

The material contribution of this work includes two algorithms, the VAP and VAP+ algorithms, and experimental results of applying the VAP and VAP+ algorithm to the concrete problems of bin picking and surface disinfection. The VAP algorithm is designed to allow for the investigation of the principles underlying the choices humans make when given the authority to select the autonomy level of a system, whereas the VAP+ algorithm implements the principles thus learned so that the autonomous system is capable of selecting the autonomy level in the same way as the human. For the VAP+ algorithm, we present a derivation and mathematical proofs of efficacy and performance characteristics, as well.

The VAP algorithm is constructed so as to implement intervention planning for a sequential task based on sorting by the cost rates of the autonomous modules, and extends the common model for variable autonomy, in which the task of setting autonomy level is assigned as a manual task to use the human's competence in evaluating environmental conditions as the central manual control element. By establishing the autonomy levels in terms of the machine module effectiveness, we effectively reduce the search space for the planning, and couple the scale to a suite of metrics which are easily and reliably measured, by which means the impact of environmental variance can be safely lumped into variances in the hidden human performance model, as the human controls the response to changes.

The VAP+ algorithm is a more sophisticated system which is able to tolerate problem

structures representing a non-sequential process model by adapting graph theoretic models for the full task to calculate a joint-cost metric and subsequently define levels of autonomy in terms of impact of replacements on the joint cost over the full task using an optimization method drawn from observations in the experimental results for the VAP algorithm.

One of the significant benefits of this approach is that the generation method for identifying and assigning the levels of autonomy also comes coupled with a metric evaluation of the autonomy level performance at each step- necessitating an iterative planning stage, but producing a known optimal level output, meaning that the human operator is obviated from the need of selecting the estimated optimal autonomy level, as well as eliminating the need to determine a human cost function to perform the optimization calculations, instead requiring only a probability estimate function, which can be readily supplied by numerous predictive systems with well-established track records of success.

For both algorithms, significant analytical proof is provided of performance efficacy and local optimality, as well as presenting evaluations of the resilience to variations and perturbations, specifically as a means to underwrite the effectiveness of the algorithms in dealing with shifts due to conditional changes- one of the most important aspects of the problem as mentioned above.

Beyond the analysis as demonstration of effectiveness, we also present experimental results which allow for the validation of the algorithms in real contexts, as well as enabling probing of some of the more particular features. One particular case of this is the relation between the VAP and VAP+ algorithms, for which results from human user control patterns is a key design element underwriting the VAP+ algorithm's primary heuristic.

These contributions are summarized below:

1) The VAP algorithm

The VAP algorithm is a method for assigning subtasks to autonomy levels in the context of a sequential task acheivement framework, using human control of a sliding scale level of autonomy as the autonomy level selection mechanism. It is useful for investigating the policies human operators are implementing in a structured context without complexities

introduced using a fully combinatorial space of planning options.

2) The VAP+ algorithm

The VAP+ algorithm is an extension of the VAP algorithm based on observations made during experiments made using that prior algorithm, which leverage patterns observed in human policies to develop a joint cost metric and a greedy assignment algorithm for generating autonomy level assignments on basis of changes in expected cost of executing the task in full.

3) Analysis of the VAP+ Algorithm

Based on the structure of the planning and assignment system for the VAP+ algorithm, we are able to perform analysis which illustrates the general local optimality of the algorthm, give conditions for the global optimality of its solutions, and establish the range of deviations from these conditions which still produce optimal levels.

4) Robot systems and Software

For validating the VAP+ methodology, we implemented two tasks on the Adaptive Robotic Nursing Assistant (ARNA) robot, which effort prompted the development of a range of beneficial technical assets for this robot, including multiple computer vision algorithms for bin picking and object tracking, robust control systems for restricted dimensional kinematics, communication protocols for sensor data acquisition, and object pose estimation and grasping heuristics.

5) Experimental Validation of VAP+

Using these assets, we applied the VAP+ algorithm to the task of optimizing human/robot system performance on two tasks: bin picking and surface disinfection. The former is the natural robotic task of picking an object from a surface and placing it in another location, while the former is the tracking and approach to a target, followed by the execution of a protocol designed to render the target sterile. In both cases we collect data necessary to implement VAP+, and then validate the performance of the algorithm against measured real-world execution efforts.

The thesis is organized as follows: In Chapter 2, I present the VAP algorithm and

experiments conducted within the VAP framework, with the data thus collected forming the basis of the subsequent work and derivations. In Chapter 3, limitations of the VAP algorithm are discussed, and the VAP+ algorithm formulated on basis of resolving these weaknesses, and proofs of efficacy presented based on this derivation. In Chapter 4, I present the Adaptive Robotic Nursing Assistant Robot (ARNA), and describe the subsystems implemented on this robot which are used in the subsequent experiments on the VAP+ algorithm. Chapter 5 presents a robotic disinfection task performed with the ARNA robot, and the results of applying the VAP+ algorithm to optimization of the joint human-robot task formulation of this task, with a focus on validating analytic properties of the VAP+ algorithm derived in Chapter 3. Chapter 6 presents results from bin-pickign tasks performed with the ARNA robot and optimized with the VAP+ algorithm, focusing on demonstrating real-world applicability of the VAP+ on a fundamental task. Finally, Chapter 7 concludes the document with a summary of work, discussion of limitations of the VAP+ algorithm, and directions for future work.

In this chapter, I detailed the motivation for, and pertinent background to the development of algorithms designed to assign tasks to levels of autonomy in human-robot interactions, with a specific emphasis on the importance of rigor to the construction of such algorithms, and with an eye towards identifying the presence of rigor in controls-based frameworks, and the relative dearth of work of the same kind in executive control systems which implement variable autonomy, highlighting the need for an approach which can provide mathematical performance guarantees about optimization of these processes.

CHAPTER 2

VAP ALGORITHM

In this chapter, I describe the motivating factors, development, and implementation of the VAP algorithm. This chapter is arranged into sections as follows: A discussion of sequential tasks as relates to automated planning algorithms and applications in robotics; the motivation for the VAP algorithm, rooted in the discussion of sliding scale autonomy as a response to the complexities of intervention planning; the formulation of the planning algorithm itself; an analysis of the planning algorithm as pertains to effectiveness; and a discussion of the means of implementation of the VAP algorithm in practical situations.

Herein, we present a novel assignment model for levels of autonomy based on the failure rate of the robot's autonomous modes in independent sub-tasks. In this model, levels of autonomy are directly correlated to the successful completion probability of their corresponding task modules. We validate this assignment model in the context of teleoperation, and propose a human-robot Variable Autonomy Planner (VAP) for applications to a state-machine control architecture. VAP elevates the human operator to a meta-level of control authority via a sliding-scale model for task accomplishment. Each task is decomposed into a hierarchy of basic actions, for which an autonomous and manual control mode are available. By changing the degree of autonomy, the operator can select the proportion of automatic action the robot will take on a case-by case basis.

By implementing this approach to identifying human-selected autonomy level plans for sequential tasks, we are able to make observations about the nature of human-selected autonomy level plans in the context of ordered inclusions sorted by cost, which later informs our work in extending the concept to fully autonomous autonomy level assignment.

2.1 Definitions

In this first context, we are presuming that an autonomous agent is attempting to proceed from a starting state S configuration to some goal state G by completing a series of K-many subtasks, P_i. In the general case, planning presume some contingent relationship between any intermediate subtask and others: a given subtask (which is not the initial state) must be reached from some other state, and must only be reachable from some set of prior states. This formalizes the notion of prerequisite tasks. A sequential task is one in which all constituent subtasks are precisely orderable: any given state may only be reached by completing one preceding subtask, and leads exclusively to a single successor subtask.

For the purposes of this research it is beneficial to consider the modeling of sequential tasks in a graph theoretic framework. When modeled as a graph, a sequential task is defined to be a directed chain. Under circumstances in which a sequential task is considered to have multiple possible results (such as failure modes) it can be considered to be a directed tree. For our purposes, however, this chapter presumes that any such branching task has a singular desired goal state and permits only one solution path from the starting state to the goal.

I will be consistently describing tasks in terms of directed graphs, and in particular as directed acyclic graphs (DAGs), with the understanding that unless otherwise stated, a task is presumed to have an acyclic model. This is a limited, but fairly generalizable problem model. In particular, an enormous variety of automation problems can be readily modeled as sequential tasks- for instance, in the industrial context, most any assembly line process is sequential. It is worth noting that such a task may not be essential so, but for the purposes of a given implementation it may be considered to have only one solution as a matter of practice (eg. an assembly line process may be re-configurable to an alternative order of actions, but on the line, it can only be executed in the order which is selected at the moment of action).

We therefor define a task as being an ordered set of states which define a directed chain from S to G: $\{S, P_1, P_2..., G\}$. Further, each state included within T is associated

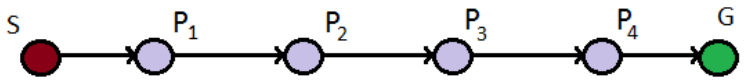

Figure 2.1: Illustration of sequential task

with some modules which allow for accomplishment of the subtask associated with that state. Note that these may include active, passive, intervention, or even null actions- the purpose here is to incorporate the diversity of state machines representing a procedure to resolve the transition from S to G.

It is convenient to represent tasks as directed graphs, which for sequential tasks, take the form in Figure 2.1. For a sequential task, the associated graph, T_G, is constructed trivially: each subtask $P_i \in T_G$ is associated with a node, with edges $\{P_i \rightarrow P_{i+1} \forall i < n\} + \{S \rightarrow P_1\} + \{P_K \rightarrow G\}$

In addition to modeling the task, it is also prudent to define a plan within the context of a task. In the context of DAGs, we can isolate natural starting and ending points- the set of nodes with indegree zero, sources, are presumed to be valid starting states: $S \in \{deg^- P_i = 0 | T_G\}$. By contrast, goal nodes are drawn from those with outdegree zero: $G \in \{deg^+ P_i = 0 | T_G\}$. A plan on T_G is defined in terms of its start an goal states, as a sequence of edges which form a chain in T_G which begins with S and terminates in G: $\{S \rightarrow P_{i1} \rightarrow P_{i2}, ...P_{iK} \rightarrow G\}$

Each of the subtasks, then, has a form: $P_i : \{(A_i, M_i)\}$, illustrating a set of autonomous (A_i) and Manual (M_i) modules. Autonomous modules are processes performed entirely by the robot, and manual modules are those requiring some level of intervention from an operator (Though it is worth noting that a manual module may still require autonomous support, such as a control system for motion operation by a joystick). From this, we can define a restricted-structure expansion of directed graph, T_G on which the planning algorithm operates by means of expanding nodes in the task graph. For each task graph node, P_i, we construct a series of nodes associated with the modules A_i and M_i. For each 'node' in T_G, a set of nodes corresponding to the modules, each with identical connectivity

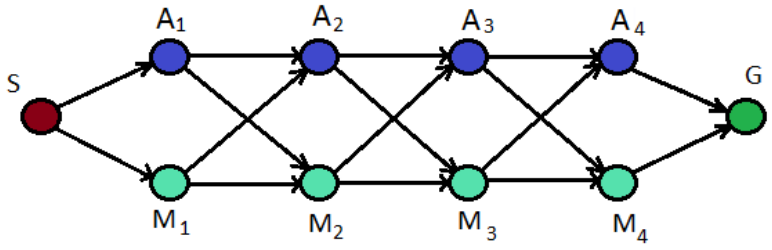

Figure 2.2: Constructed planning graph for a sequential task in which each subtask contains one human-driven and one machine-driven module.

to the parent node P_i is constructed, such that x, $\forall A_i, M_i \in T_G : \{x \rightarrow y | y \in P_{i+1}\}$. Put otherwise, every module in T_G has an edge to the next module in P_{i+1}. Such a construction is illustrated in Figure 2.2.

While this does represent an increase in complexity for the assignment algorithm, the restricted structure also allows for some readily observable properties. Most immediately, for any sequential T_G, all task plans follow a strict progression- there are 2^K possible execution plans, but the modules are always executed in index order. This naturally presents a significant reduction in planning complexity of course, but also represents an even more significant factor: for each subtask, the essential planning decision is between the M_i and A_i modules.

The general problem of planning on a sequential task is, of course, trivial- there is only one transition available at any point, and thus no decision problem embedded in the problem model. However, we are specifically addressing the context of human intervention. As such, we must make a distinction between the solution to the *task* and the *autonomy level plan*. The sequential task model is presumed to be a chain, always, but the assignment of autonomy levels for that task may include a restricted class in which parallel means of achieving a state transition are available at each task state, vis-a-vis T_G. We thus define an autonomy level plan, ρ, as a structure which represents which subtasks are to be executed manually and which are to be executed autonomously. For instance, in figure 2.2, a possible such autonomy level plan is $\rho = \{A_1, M_2, M_3, A_4\}$, indicating that P_1 is executed autonomously, P_2 and P_3 are executed manually, and P_4 is also performed autonomously.

Further, to guide the selection of autonomous and manual modules, we select some

cost function, μ, on which planning optimization occurs, where in we may compare the utility of modules by a value $\mu(A_i)$ or $\mu(M_i)$.

In this section, we define a **cost**, μ, such as completion time, probability of success, operator fatigue, etc. which determines relative efficacy of task performance. $\mu(M_i)$ represents the cost associated with the manual module associated with the i^{th} subtask, $\mu(A_i)$ the same for the autonomous implementation. We will posit a selection of required properties of the cost functions which can be used along with the VAP algorithm. These properties will be needed later to ensure that the cost functions are cumulative, aggregate, and mutually independent across a task graph.

Let X, Y, or Z to be arbitrary modules of either manual or autonomous type. Additionally, $\circ$ is the operator representing the method of combination of multiple μ costs into a joint cost - for instance, with the example solution in the previous section $\{S, M_2, A_4, G\}$ as in Figure 1, the net cost would be given by $\mu(\rho_{A_L}) = \mu(M_2) \circ \mu(A_4)$. The cost function satisfies the following conditions:

Condition 1: Costs are non-negative, as otherwise it may be possible to construct an autonomy level plan with infinitely decreasing cost; $\mu(X) \geq 0 \ \forall X$.

Condition 2: When combining costs, the joint costs are monotonically increasing or decreasing when adding more subtasks; $\mu(X) \circ \mu(Y) \geq max(\mu(X), \mu(Y))$ **or** $\mu(X) \circ \mu(Y) \leq min(\mu(X), \mu(Y))$

Condition 3: The cost function is transitive, ensuring that the effect of a module's independent cost on the cost of any autonomy level plan is the same; $\mu(X) \circ \mu(Y) = \mu(X) \circ \mu(Z) \rightarrow \mu(Y) = \mu(Z)$

For example, the cost function associated with human-robot tasks μ could be represented by task execution time, while the operator $\circ$ is addition. Indeed, times are measured in positive real quantities, are strictly increasing, and inherently transitive in a task sequence. Another example of a cost function is task probability of success, combined under the $\circ$ operator of multiplication. This cost function is also positive and transitive, but the cost of a task sequence will be strictly decreasing. Evidently, though the conditions may

seem restrictive at first, it is clear to see that it is possible to identify many very frequently used such measures which are in alginment with these requirements. For instance:

I. Cost

In general, cost functions are presumed to be iterative and increasing, while being subject to minimzation. However, because we have presumed that the planning graph is a DAG, with module selection restricted to each subtask, it is also possible to optimize for maximal additive cost gains in this context- hence the absence of a min/max criteria in the conditions for mu. Further, we might hypothesize an exponentially scaling cost gain function, such as one in which production levels at one step determine the production limit at a latter step- in this case, the gains may be multiplicative, and this too is optimizable for a sequential task. Even more important, though, is the recognition that accommodating a multiplicative gains model for such a task allows parametrization of a task which, under assumed linear metric models, could not be cast in the form of a sequential task, as the impact of latter production would imply a branching path.

II. Time

Time is a natural and common metric for performance, and in general is presumed to be a minimized quantity under additive composititon, however the same precepts discussed above as regards to cost are also applicable to time, though often time constraints are subject to strict linear progressions rather than geometric growth, especially when condition 3 is applied to natural systems. Despite the more restricted modeling application, though, the potency of the objective function conditions still enables analysis of formerly branched-potential applications as sequential tasks with a suitable composition function.

III. Probability

Finally, probability represents a divergent optimization function from the common application, however in adopting the variant form in condition 2 we are able to utilize net likelihoods for optimization. Because joint conditional probabilities are typically measured

under an assumption of independence, they naturally meet condition one, and under composition by product meet the latter two conditions as well. This allows us to determine an optimum under maximization across the graph, essentially determining a path with the maximum probability of reaching the goal state. Probability is a particularly useful metric when considering failure versus success of a system wherein the output quality is a binary function.

One might immediately notice that these are simple constrictions on the definition of a metric function, adapted to be expressed on a graph and with the constraint that the function be path independent (by condition 3). These properties are meant entirely to constrain our objective function to policies which are monotonic under composition and positive definite and, as such, subject to optimization under the Bellman conditions. We are not going, at this point, to attempt to qualify optimal performance under this criteria, rather, we want to ensure that an optimal condition can be identified, with the aim of implementing human-based autonomy assignment selection occurring, such at we can study this selection and identify the pertinent objective function the operators are intuiting.

If such a suitable function is available for a specific planning problem, then we may consider a number of potential means of optimizing the path through a graph. There are of course many available path-planning algorithms, however in this case the structure of the graph makes the selection uncommonly simple- the optimal performing member of each set of subtasks, in their order withing T_G, is necessarily the optimal plan.

It is possible to see this based on the combination of conditions for the objective function and the observation that each step in the plan must correspond only to the same-indexed member of the task sequence, as noted in the prior section. Consequently, we can write a simple optimization principle:

$$\rho[i] := argmin_i[\mu(P_i) \cdot \mu(P_{i+1})] \tag{2.1}$$

if $\mu(x) \circ \mu(y) \geq 0$, or alternatively:

$$\rho[i] := argmax_i[\mu(P_i) \cdot \mu(P_{i+1})] \tag{2.2}$$

if $\mu(x) \circ \mu(y) \leq 1$

And thus, by the assumption of condition 3 in μ, and the constriction of each decision to the discrete subtask, thus automatically satisfying the principle of subproblem optimality, letting us know that coherent optimizable objective functions must exist, though the explicit determination of such is subject to μ and $\circ$.

With this particular formulation, the restricted class of graph structures available for intervention planning suggests that a specific set of algorithms may be particularly effective in implementing intervention. In particular, depending on the selection of a suitable metric function μ, greedy sorting algorithms present a viable, computationally simplistic, avenue to effective planning.

All these models, however, have the fundamental presumption that μ is measurable and well defined for all modules under consideration. This presents a significant problem when uncertainties arise- from an analytic standpoint, the biggest problem is the issue of condition 3, which establishes the comparability of the independent steps. If we were to relax it, then the ordering comparison which allows condition 2, the composition constraint, to operate on each subtask independently no longer applies. For instance, if there is a probability function $\delta\mu(P_i)$ representing the distribution of values, centered at $\mu(P_i)$, then if:

$$\delta\mu(P_i) \cdot \delta\mu(P_{i+1}) \geq \mu(P_i) - \mu(P_{i+1}) \tag{2.3}$$

the probability that P_i and P_{i+1} are mislabeled in the autonomy level plan with respect to μ, assuming w.l.g that $\mu(P_i) \leq \mu(P_{i+1})$ is given by:

$$P(Err|P_i, P_{i+1}) = \int_{min(\mu(P_{i+1}))}^{max(\mu(P_{i+1}))} \int_{min(\mu(P_{i+1}))}^{\mu(P_i)} \delta\mu(P_i)\delta\mu(P_{i+1})d\mu \tag{2.4}$$

In general, the expected loss would then be $P(E|x, y)(\mu(x) = \mu(y))$. This loss can be considerable. For instance, if we assume uniform distributions of uncertainty, as

well as a Gaussian distribution of actual metric values, then the expected loss would be approximately equal to the variance among the metric value measurements, meaning the planning algorithm would be only marginally better than random guessing.

As such is the case, an objective function for selecting relative values which account for uncertainty with a bias which maximizes the skew between different values is ideal. This property is what suggests the value of sliding scale autonomy for practical implementation of an intervention planning system in a real-world context. While aggregate machine observations are easy to record, the derivation of a fully autonomous planner which is able to convert environmental data into task plans for which uncertainty is skewed towards the separation of metric states is, as discussed in the previous chapter, a monumentally difficult problem. However, our goal with autonomy level assignment is to optimally leverage the contrasting competencies of the human operator and the autonomous agent, and in this case, it is apparent that humans have superior capability in judgement as regards environmental variance. This is the essential principle behind adjustable autonomy- using the operator as the system to determine the bias factor in the decision problem.

This re-casts the problem from one of analytically determining the selection criteria to one of implementing a metric adjustment which is natively suited to integrating with human faculties. Many such systems are extant, however the particular formulation of the problem as we have outlined here presents an opportunity to develop a rigorous, mathematically stable model for the sliding scale. In defining the metric-based optimization problem, we have incidentally cast the objective function from a functionally $O(2^n)$ scaled decision problem in n sets of branching choices to an $O(n)$ decision problem across n sets of two modules each. This linearization then needs only be ordered by μ to be converted from a decision problem to a scale function. This concept: implementing sliding scale autonomy as a metric-based ordering of decision problems, is the kernel of our system.

2.2 VAP Algorithm Formulation

In applying this principle of optimization, we will structure the algorithm by returning to the optimization principles defined earlier and connecting them to levels of autonomy. For any sequential task, replacement of any machine-driven module by a human-driven module clearly represents a change in the system's level of autonomy, but a replacement of any one module out of the many options may lead to differing levels of system performance. Rather than attempt to order the many nested possibilities available for sorting all binary combinations ion the set, we instead elect to couple the autonomy level to the system performance such that each change in autonomy level leads to a strictly monotonic change in expected performance.

This alone is not sufficient to fully define order-able levels, but the aim is also to alter the bias function such that uncertainty in planning order is also reduced. Because of this, we can look to the means by which we can measure the performance metric, and choose the ordering principle in such a way as to provide a scale which is correlated to the metric itself. As a rule, the performance of the machine components will be significantly more certain than those of a human. Firstly, because machine performance is more measurable, and second because it is more repeatable. If we could, then, formally eliminate the impact of human uncertainty, we would fundamentally shift the expected loss due to uncertainty towards the variance of the machine system.

To use this principle to generate a scaled set of autonomy levels and from them define our autonomy levels, A_L. A_L is defined as a parameter in $(0, K)$, for K+1 total levels, which describes the number of subtasks which are executed autonomously. We can simply elect to use the value of the machine module metrics as the scaling factor by which autonomy level assignment is made, as we are already sorting by this parameter. Suppose that we sort the modules by $\mu(A_i)$: $A_{list} = [A_{j1}, A_{j2}, ...A_{jn} | \mu(A_{jl}) > \mu(A_{j(l+1)}) \forall l]$. We then define the level of autonomy as the binary inclusion or exclusion of machine modules based on their order in this list, with the autonomy level be represented as A_L being the threshold, then each level is thereby defined as a plan constructed by the objective function:

$$\rho_{A_L}[i] = \begin{cases} A_i & , i < A_L \\ M_i & otherwise \end{cases} \tag{2.5}$$

Defined across the entire plan, then, the joint cost across the plan can be expressed as:

$$\mu(\rho) = \prod_{i=0}^{A_L} p(A_i) \cdot \prod_{i=A_L+1}^{K} p(H_i) \tag{2.6}$$

For probability and other decreasing μ, and by contrast, for time- or cost-based functions:

$$\mu(P) = \sum_{i=0}^{A_L} C(A_i) + \sum_{i=A_L+1}^{K} C(M_i) \tag{2.7}$$

This plan, then, contains the A_L many best-per forming A_j, with all remaining subtasks being performed by manual modes. It is trivial to demonstrate that each autonomy level thus defined possesses the best performance expectation for the given number of autonomous modules selected. Were any of the M_j modules in the plan replaced with other machine modules, the aggregate performance of the autonomous components must decrease, as only less-well performing modules are available. By conditions 1 and 2 for μ, this means that there is a necessary reduction in the efficacy of the autonomously performed components of the system.

In practice, use of the VAP algorithm is primarily a process of system design for shaping a simplistic augmentation to a hypothetical extant system (namely, the known optimizable system as described via the cost function conditions previously). Usually, we presume there is a system implemented in such a way as to be represented as a state machine (as alluded to by our initial graphical representation of the task, T_G). From this initial state machine framing, in which an arbitrary set of autonomous modules, necessary manual modules (if any), and modules which bear potential replacement is present. The

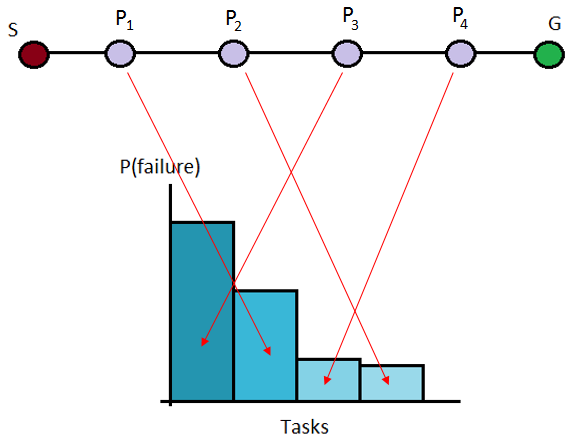

Figure 2.3: Sorting the task module sets by failure rate of the autonomous modules

VAP algorithm itself only controls the segments of the state machine in which replacements may be made.

Application proceeds as follows:

Step 1: The failure rates for each autonomous module are determined experimentally. Note that this will be a context-dependent empirical measure, and should reflect, roughly, the average performance over the expected application context.

Step 2: The appropriate (H_{ij}, M_{ij}) pairs are composed and sorted by the respective failure rate. The order in this list determines r_{ij} for each pair. (Figure 2.3)

Step 3: During execution of the task, the primary planner, P, takes state information from the robot and determines which task to perform next, indicated by the index k.

Step 4: Prior to beginning any task, the operator my, by some interface, change the value of the A_L parameter, in accordance with their judgement of the problem complexity.

Step 5: The level of autonomy is then checked by the augmented planner (Algorithm 1). If the rank of the current module pair is lower than A_L, then the human-interfaced module is used, and if the rank is greater, then the autonomous module is used.

Applied as such, the VAP therefor selects whether the autonomous or manual module is used for each subtask as they are performed. With each autonomy level, failure probability is used as the relative ranking metric, thus as autonomy level decreases, increasingly successful automatic modules are discarded in favor of manual ones. Put otherwise,

an increase by 1 unit of A_L corresponds to replacing the least successful machine module in-use on the prior level with a human operated module.

Below in Algorithm 1 is detailed the application format of the algorithm itself, in all its simplicity, where s_t is the current state, A_L the user-set autonomy level, k the index of the next subtask, and r_i and r_k the ordered index of the current- and next-indexed module sorted by probability of failure of A_i.

Algorithm 2.1 Variable Autonomy Planner

Input: (s_t, A_L)
Output: (s_t, A_L)
$k \leftarrow \rho[s_t]$
$r_i \leftarrow r_k$
if $r_i < A_L$ **then**
  $\rho_{t+dt} \leftarrow M_k$
end if
if $r_i \geq A_L$ **then**
  $\rho_{t+dt} \leftarrow A_k$
end if
return ρ_{t+dt}

This acts as a direct augmentation on an existing state machine-type control system, in which at each step wherein a selection between autonomous and manual modes may be present, and directs the robot to execute one of these two potential steps on basis of the externally set A_L which is, naturally, a feature of the user input.

2.3 Robotic Bin Picking Experiments with VAP

To explore an application case of the VAP algorithm, we implement a case study in a robotic system accomplishing a simple task. We experimentally investigate the effectiveness of this algorithm by testing 15 subjects using a robotic pick and place task. Task performance was measured by completion time and failure rate. Our experiments indicate that in most cases, fixed-autonomy levels have worse over-all performance than the case in which our VAP algorithm selects the autonomy level. By contrast, when operators are allowed to select the level of autonomy, they can make an educated assessment of the problem and select superior balances of speed and quality through module choice.

In this chapter, I will describe the picking task to be performed, including a discussion of the Rethink Baxter robot, the software capabilities we have designed to accomplish this picking task, the decomposition of the sequential process used to accomplish it, the experimental procedure for acquiring data, and finally present and discuss the results.

2.4 Task Description

To address the cases in which the utility of the GAP algorithm is significant, we turn focus to a specific task in which the conditions for application are present: sequential subtasks, readily observable metrics, and an inverse relationship between autonomous and manual performance.

In particular, this last condition, as a means for exploration of functionality of the concept, is one we can enforce by careful construction of the task and the autonomous modules. In the prior section we discussed this technique as a theoretical concept, and in this case we approach the problem of designing a system implementing these principles in addition to the simple chore of validating the system.

Because it meets the abstract requirements, presents a straightforward use, and has a large body of work which allows ready performance comparisons (admittedly the body of work from which the conclusion of test suitability was drawn), pick and place is our sample problem of choice. These reasons will be explored in more detail in the technical discussion below.

2.5 Robot System

Because the hardware system on which the task is actually performed sets fundamental boundaries on performance and the methods of use, we describe it first, here.

For these experiments, we implement pick-and-place functionality on a ReThink Robotics Baxter Research Robot, as shown in Figure 2.4.

This robot is a fully integrated system, including built in kinematic and inverse kinematic libraries, subroutines for controlling peripherals, and an interface to ROS, the

Figure 2.4: Image of the ReThink Robotics Baxter

robot operating system. The full robot includes two 7-DOF arms, a two-finger gripper on each as an end-effector. In our experiment setup, we are using only one of the two manipulators. These manipulators are controlled programatically via a remote basestation computer running ROS, from which commands are issued via ROS topics to subsystems instantiated on the onboard CPU within the Baxter. This remote workstation provides the primary interface for the system.

With regards to safe operation, the Baxter unit employs a fully back-drivable set of actuators in its manipulator arm, meaning that the robot can be easily manipulated by external force, such as if an operator needs to resist its motion in a collision. This is helpful especially in collisions with environmental obstacles, obviating the need for sophisticated collision planning and intervention where simple checks can suffice.

The Baxter is also equipped with a wrist-mounted camera. We use this camera for the recognition and detection of objects in the workspace. In addition, we provide a raw video stream from this camera from this sensor to the user's teleoperation workstation during manual positioning and grasping contexts, to augment the operator's perspective, which may be limited by distance from the robot's workspace. The user interface can be seen in Figure 2.5.

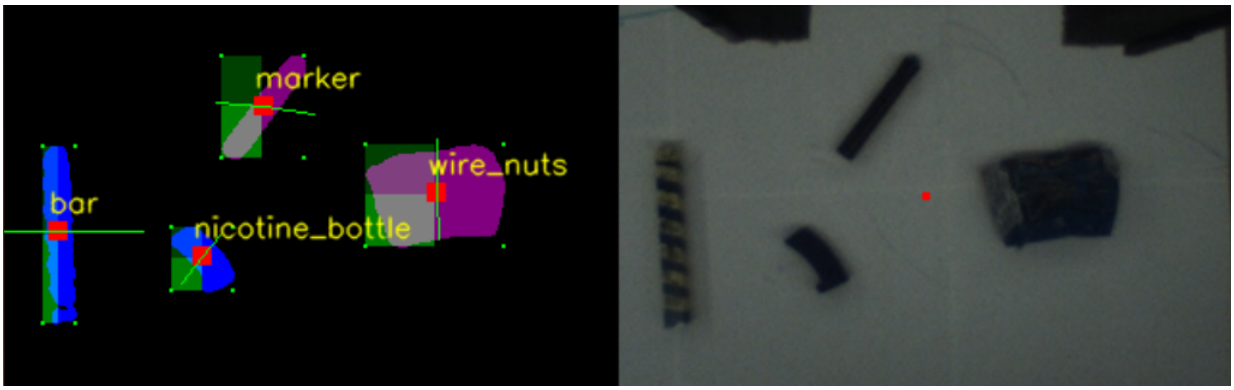

Figure 2.5: Image of the user interface implemented for manual control of the Baxter during the experiments, the left panel showing the object classification system, image filter, and angular approximation for gripper orientation; the right panel shows the base wrist-camera frame in which the user observes the workspace.

2.5.1 Picking Task

Pick and place tasks present a reliable and well explored space in which to test our precepts- the variety of techniques available for each component subtask allow for selections to be made enforcing a high degree of regularity on performance, and systematic implementations of teleoperation control to be matters of integration rather than development. The goal of this task is to collect a specified item from its resting place on a table within the robot's functional workspace using a robot manipulator arm with a gripper. Items are initially on surface in random poses, and grasping is achieved in our case by a vertical orientation of the gripper, for which the yaw about the vertical axis allows for finger alignment.

As with the theoretical investigation of the GAP algorithm, the first requirement is the task decomposition. We decompose this task into the following sub-task domains: Position Determination, Orientation Estimation, Manipulator Placement, Object Identification, and Object collection; this group of processes defines six levels at which we can implement variable autonomy.

1. Full Autonomy: Wherein all actions are performed without human input
2. User sets XY Position: In which the operator specifies the 2D planar coordinates

of the item

3. User sets Yaw Control: In which the operator specifies the angular orientation of the gripper

4. User sets Object Identification: In which the user identifies the object within the visual field

5. User sets Object ID & Yaw Control: In which the user specifies the object and sets the grasp angle

6. User sets XY Position & Yaw Control: In which the user provides exact coordinates and grasp angle

Note that for this task decomposition, not all tasks have fully manual or fully autonomous modes individually, but the tasks do still proceed in a strictly sequential order. While the adherence to other conditions- specifically the inverse relationship between autonomous and manual modes- for efficacious operation of the VAP algorithm are not clear, we will see in the data that these precepts hold well.

For testing, we investigate three different scenarios: single object detection and picking; multi-object picking; and cluttered environment picking. In each version, the task is to pick an item from the table. All cases are tested under both fixed-mode and variable autonomy, allowing us to develop the full set of performance curves across human and machine performance by performing a panel across all autonomy levels and complexity classes. The complexity class change allows us to examine the patterns in operation of the sliding scale as environmental conditions change, with the induced complexity changes acting as a study across the autonomy levels under increasingly adversarial prompts to the efficacy of the autonomous modes.

We also define two performance metrics: completion time and failure rate, where failure is defined as losing control authority over the picked item at any point in the trial- in particular dropping the item, though failure to acheive a grasp also counts as a failure as well. In cases in which an autonomous system performs object recognition, a misclassification also

constitutes a failure, as there is no causative means for the system to collect an incorrectly classified item in the singular or multi-item cases. It is possible to accidentally grasp the correct item when seeking an incorrect one in the clustered item case, however this does not truly represent successful task completion.

2.5.2 Experiment Procedure

Our trials proceed as follows: twelve subjects are tasked with completing each of the three different problem domains discussed above: single-item, multi-item, and cluttered pick-and-place. Each subject completes six trials in the fixed mode cases, as well as two trials using the VAP setup, for a total of four trials per A_L in the fixed mode for each problem complexity class, and eight trials per complexity level in the VAP-selected case.

In fixed mode trials, the most-failing autonomous modules are replaced with manual operation segments driven through the UI; in the sliding-scale tests, the user is allowed to dynamically adjust the autonomic components between trials. Four total items are presented to the robot, as shown in Figures 3 & 5, and these items are randomly placed and aligned. Additional objects are used for the cluttered test, but not manipulated. The manipulation requests are limited to the specific four items shown on basis of the object recognition machine module, which implements histogram-matching, chosen specifically to create a distinguished success rate at a level which occasionally warrants human intervention (as discussed below)

In each trial of the above experiments, we record both the number of times the task fails and the time to completion, representing accuracy and speed respectively. Our analysis hinges on examination of the impact of the A_L parameter on these two metrics in the fixed-mode trials to support our assertions regarding the relative merits of the robot and human's capabilities, and on the comparison of average performance characteristics of the variable autonomy case versus the fixed-mode case.

A block-diagram of the primary process which implements module-exchange between the autonomous components and the manual components is shown in Figure 2.6. The level

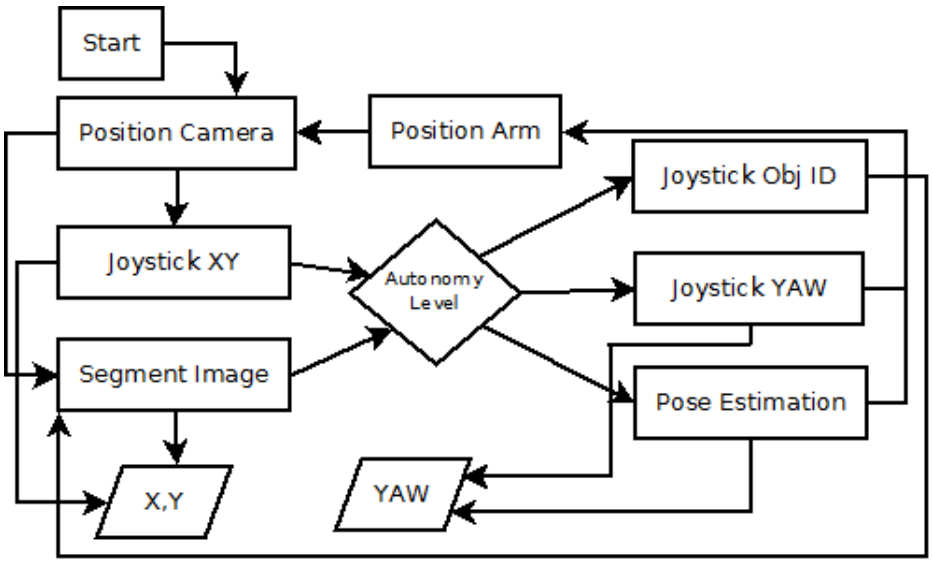

Figure 2.6: Main Software Diagram

of autonomy acts as an additional state variable for P. In our software, the VAP is thus introduced precisely as a new variable to the pre-existing state machine, for which the functional block is contains Algorithm 1.

For the autonomous modules, we have constructed a fully independent pick and place system, comprised of the Baxter's ROS based inverse kinematics and joint position control modules, and machine vision perception components leveraging OpenCV bindings for python. To implement object detection, a background filtering method was used, with segmentation based on marching squares blob-detection. Pose estimation is performed by comparison of bounding box area to object area. Tracking estimates are made by calculating the visual center of mass of the object, and projecting pixel coordinates onto the workspace. Figure 3 shows diagnostic output of these tracking systems (left image). Object recognition is implemented using a normalized histogram comparison, which compares objects to single-frame filtered images of the objects.

The XY positioning and Object Identification components are identical in appearance to the user, except for prompting. Both instances provide a real-time feed of video from the wrist cameras, and allow the user to position a target indicator in the field of view as shown in Figure 2.7 (Left). In the XY positioning case, the registered point is converted

Figure 2.7: User interface: XY/Object ID GUI (Left), Yaw selection GUI (Right)

to workspace coordinates, and in the Object Identification mode, the nearest located item found in the visual field is tracked item for retrieval.

The second mode, as seen in Figure 2.7 (Right), is the Yaw-set mode. In this module, the user is able to set the orientation of the manipulator. A line is projected onto the video feed, and when the user twists the joystick about the vertical axis the line rotates about a fixed point at the top of the image. The projected orientation of the line matches the orientation of the gripper after rotation.

There are, in this state machine, three modes in which user input is taken from the operator: Object Identification, XY Positioning, and Yaw orientation. Each of these is driven by input from a USB Joystick, interfaced to supply X and Y motions, Yaw rotation about its main axis, and button input from its trigger. The joystick used is shown in , with labels for the local X, Y, Yaw control coordinate frame.

2.5.3 Results & Discussion

In this section, we detail the data collected during the experiments described in the prior section, and analyze the results. We can examine the relationship within the fixed-mode data as shown in Tables I & II, which describe the average task completion times and failure rates as a function of the A_L parameter. These data are also represented graphically in Figure 2.9.

The first observation we can make is that there is a significant difference in task completion time between fully-automatic execution and execution with manual components.

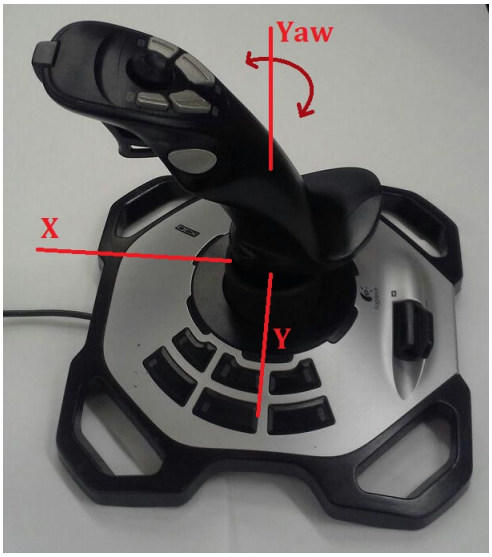

Figure 2.8: User interface: XY/Object ID GUI (Left), Yaw selection GUI (Right)

Setting	A_L: 1	2	3	4	5	6
Single	32 (0.6)	58 (2.2)	56 (6.9)	55 (9.5)	56 (6.5)	61 (7.6)
Multi	34 (1.5)	64 (10.9)	49 (5.6)	59 (7.6)	65 (2.6)	58 (7.1)
Cluttered	29 (1.7)	51 (3.1)	56 (3.2)	55 (2.3)	52 (7.9)	61 (6.9)

TABLE 2.1

Completion Time for Fixed-mode Trials: average(variance)

The average completion time over all autonomous cases (Table I, column A_L: 1), is 32 seconds, whereas the completion time average over $A_L > 1$ is 57 seconds; a 78% decrease in speed per item. However, comparing trials involving any degree of manual intervention, we find that the range of completion times do not vary significantly, with the average deviation over all fixed mode settings and complexity classes being 5.4 seconds (10.3% of the bulk mean), and the largest recorded deviation being 10.9 seconds. The abrupt change in speed between autonomous and intervention modes, without significant deviation among intervention modes, offers support for our proposition that human interventions tend to be more time-intensive.

There is also a small advantage in accuracy for our algorithm. For both the single-

Setting	A_L:	1	2	3	4	5	6
Single		10%	5%	0%	5%	0%	0%
Multi		10%	10%	0%	5%	0%	0%
Cluttered		45%	20%	10%	15%	5%	5%

TABLE 2.2

Failure Rates (Fixed-Mode Trials)

and mixed-item cases, the failure rate for $A_L = 1$ is 10% (shown in Table II). If we average over the same cases for the manual modes, we find failure rates of 2% and 3%, respectively. In the cluttered case, this pattern is even more pronounced- the fully autonomous mode's failure rate 45%, and the fault rate curve over the level of autonomy scale decreases as autonomy increases, down to 5% for fully manual operation.

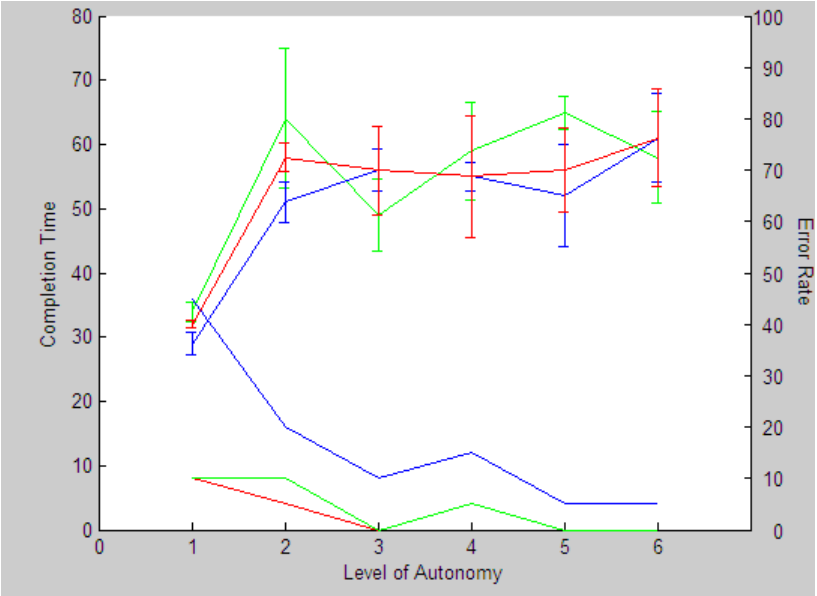

Figure 2.9: Comparison of failure rate and average completion time versus level of autonomy for all task domains (single-item in red, multi-item in green, cluttered in blue). Error bars represent sample variance, included for the completion time data.

This relationship is illustrated more succinctly in Figure 6, which plots both failure rates (right hand axis) and completion times (left hand axis) on a common graph. These experiments illustrate that the conditions of our hypothesis hold- that autonomy and intervention are complimentarily related in these two metrics.

Another important piece of confirming evidence is towards our proposition that we

have proposed tasks exhibiting differing levels of complexity. The failure rates for the single- and mixed-item cases are very similar, but those for the cluttered case are uniformly higher. This verifies that we are indeed changing the context of the test cases over these problem classes, in the sense of presenting an increasingly challenging objective to the human/robot system.

Our results also confirm that the best performance is achieved through a shared-control model: in the single-item case, we find that $A_L = 3$ is the level at which the failure rate is lowest (0%) and the speed is highest, and likewise for the multi-item case. For the cluttered case, however, the highest-speed/least-error performance is achieved at $A_L = 5$. In none of these trials was the optimum performance achieved in either the fully autonomous or the fully manual mode.

As the complexity of the task increases, we see a shift in the level of autonomy associated with the optimal performance. This indicates that the ideal A_L is indeed context-dependent over the chosen experiment domains. By examining the performance of the Variable Autonomy Planner in Table III, we find that the cumulative error rate is reduced to 0% in the single- and multi-item cases, and 5% in the cluttered item cases.

Additionally, across all three domains, the variable autonomy method exhibits marginally faster completion time than in the fixed-mode case, though still less quickly than the autonomous case. On average, this difference amounts to a 9% speed increase over the corresponding fixed mode case, and a 5% decrease relative to the autonomous speed. This is outside the variance for the single- and multi-item cases, but within it for the cluttered case. Determination of whether this discrepancy is an artifact or a true pattern is a topic

Setting	Faults	Time per item	Favored A_L
Single	0%	50s	3
Multi	0%	46s	3
Cluttered	5%	51s	5

TABLE 2.3

Variable Autonomy Results

of ongoing experimentation.

In the single- and multi-item cases, users primarily selected to the Yaw setting ($A_L = 3$), and in the cluttered experiment users tended to select Object Identification and Yaw control ($A_L = 5$), leaving only XY positioning to the perception system. In the fixed-mode cases, A_L's are the same as were determined to be the sought-after optima, indicating that the users were able to select the optimal autonomy level. Because the user-selected level both changes with problem conditions and matches the fixed-case optima, this constitutes evidence that the user constitutes a good judge for selecting the A_L parameter dynamically in response to changing task conditions.

In this chapter, I presented the formulation of the VAP algorithm discussed its utility as a study system for variable autonomy under the paradigm of sequential task optimization, and demonstrated its deployment on a robotic picking task. I further discussed the implications of the observed behavior of the humans guiding the selection of the autonomy level parameter as regards the development of further, more sophisticated systems for implementing Pareto optimal selection of autonomy level in situations in which there are trade-offs between human and robot performance.

CHAPTER 3

VAP+ ALGORITHM

3.1 Limitations of VAP algorithm

While the VAP algorithm is both straightforward and effective, there are some particular weaknesses that make it unsuitable for general application. The most glaring of these is the restriction to sequential problems. While a large proportion of common robotic tasks are in fact expressible as sequential tasks, there are many, many problems in which branching possibilities in execution completely forbid application of a feed-through type process tree, and thus do not remit solutions under the VAP algorithm framework.

One useful example case to consider would be the hypothetical toy problem of an assembly task, such as that illustrated in 3.1. In this situation, the non-sequentiality comes from the potential for assembly operations potentially taking place in many orders, with multiple paths leading to the goal state.

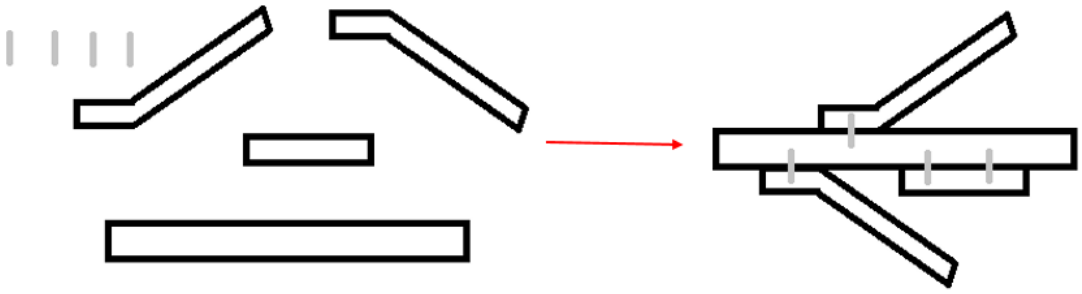

Figure 3.1: Example assembly task, in which the figure on the right must be constructed from the components on the left

It is possible to suggest that one might pick a single optimal series of choices and then

apply a sequential algorithm to that sequence, however this first presumes the existence of a planning algorithm to determine the sequence, but the second, more significant problem is one of functional pragmatism- what happens when, say, the fully autonomous and the fully manual plans have differing series of steps? Even more complex is the potential that, if we define some kind of autonomy level description which defines multiple levels, each level may have an independent optimal path.

Illustrating this in 3.2 are different some different sequence options of a procedural assembly, with the list on the left highlighting potentially costly actions for an autonomous assembly agent to perform, particularly that there is an expected high-cost to performing action 2 prior to action 3. We can imagine simply that there are many cases in which the existence of one specific step with a high cost might eliminate the possibility of any plan including it.

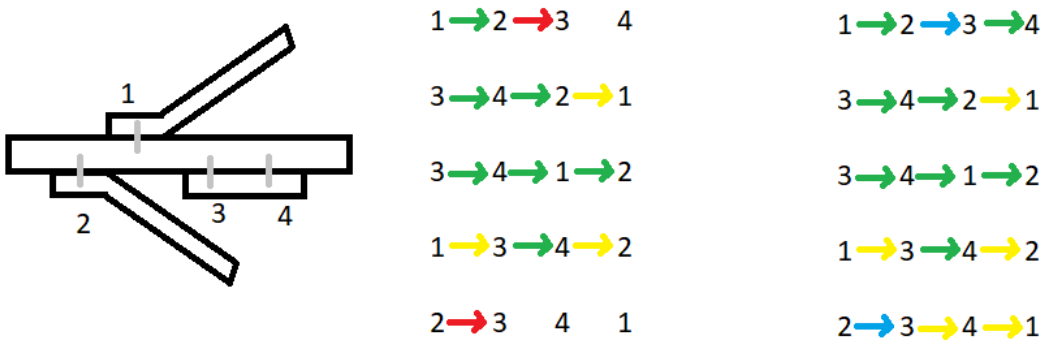

Figure 3.2: Multiple completion paths, which potential high-cost prohibitive actions highlighted

Critically, though, a human may find it relatively easy to perform this transition, and thereby open up a potentially much more efficient set of paths by intervening in this one step. Because of the fact that certain early interventions may expose other, more efficient solutions which are not pragmatic presuming fully autonomous operation. Given this potentiality, it becomes clear that a means of planning intervention on problems presenting more complexity than can be expressed in a sequential formulation is valuable.

To put the issue of why adapting the VAP algorithm to non-sequential tasks into

a more general phrasing, we can look more specifically at the notion of path optimality potentially varying across different autonomy levels in the context of defining the autonomy levels through VAP. Suppose that we do define the autonomy levels in the same way- by sorting autonomous modules by metric performance. It is certainly possible to do so, but then consider the action of the sliding scale autonomy in algorithmic application: if a first autonomy level effects the removal of an autonomous node which is not on the most effective path from the starting state to the goal, then there will be no operational difference between the first and second level.

In and of itself this is not, per se, a critical problem, as we might propose to simply define autonomy level clusters across paths, but the wide range of possible distributions of order inclusions possible mean that it is entirely possible that multiple different sets of exclusions might change the optimal path multiple times, under which circumstance it is possible that the fundamental assumptions of correlation between each level change and the total cost over the plan does not hold: when we add a new level of intervention, it effects only a subset of potential solution paths.

It is readily possible to visualize this effect by examining the impact of the VAP definition of autonomy levels when acting on a simple non-comparable task structure, as indicated in 3.3. In this illustration, we can see how certain choices of exclusions lead to problems in which individual paths develop disjoint group effects under irregular removal, and thus the net impact on the cost over the full plan is inconsistent; a result of the fact that while the total cost in the whole task graph my change, only one actual plan can be executed.

3.2 Hierarchical Tasks

The issues described in the prior section are inherent complications to working within the domain of hierarchical task planning, and are case-specific manifestations of common problems in planning and learning systems which relate to the nature of interleaving and dependent structures. For our purposes, a hierarchical task can best be described as a task

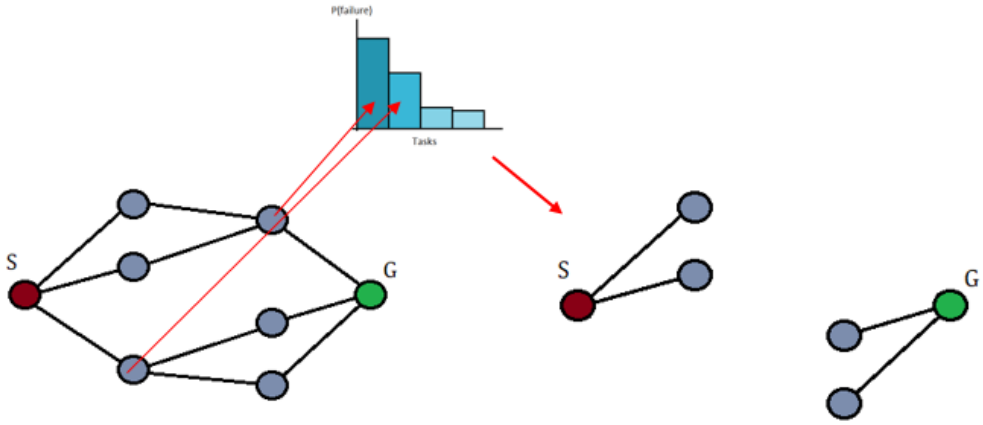

Figure 3.3: Illustration of process plan discontinuity resulting from application of VAp algorithm principles to a non-sequential task space

plan in which there are contingent relationships between ordered subtasks (just as with the sequential task) and also more than one path from the starting state to the goal state (the key distinction). Seeing this, it becomes clear why the definitions and analysis of the task performed in Chapter 3 was expressed in terms of directed graphs: they are sufficient to represent tasks in the abstract, including both sequential formulation and hierarchical, and also support the modifications for intervention plan spaces as we have used them.

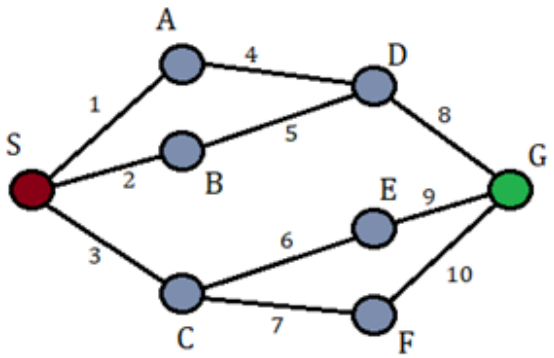

Figure 3.4: Exemplar case of a task graph for a hierarchical task, with modules labeled A-F, the start state S, goal state G, and connective edges associated with costs 1-10

Figure 3.4 illustrates a simple, but archetypal, example of a hierarchical plan which cannot be further decomposed to a sequential system. In these cases, rather than addressing

the nodes of the graph by index, we must do so by label. Further, we must also note that while the edge space was, in the sequential system, entirely defined by the nodes, such is not the case for a hierarchical problem, as can be seen by examining node C in Figure 3.4.

In these models, we may use the same definition of a plan as used for evaluating the VAP algorithm: a chain proceeding from S to G in T, but with the caveat that in this case the plan still represents a disjoint subset of M_i and H_i, but is not complete in T, as are sequential plans. That is to say, a plan on a hierarchical subset does not visit all nodes in T, only a chain within the representative graph leading from S to G.

To support analysis of these systems, we must develop a metric function which is representative and valuable across the wide range of differing graph structures we might adopt. We can make a narrower selection by way of evaluating some of our observations from the results presented in Chapter III, wherein we were able to observe a specific trend towards optimizing on the most likely to succeed member of a class with the best associated cost. To bridge these two categories together, we can define a joint cost function which is the *expected* cost, expressed in terms of cost and probability of success:

$$C_N = p_i c_n \tag{3.1}$$

Or, the expected cost of traversing edge N, literally performing the subtask execution module associated with node n, in our framing, is the product of the probability of succeeding with the measured expense of taking the step. This function provides an enormous amount of utility- first, it combines probabilistic measures and cost functions into a unified expression which is internally consistent. Second, it allows for any combined objective function to be supplemented for c_n, meaning that an arbitrary objective function containing time, monetary, and human effort losses might all be rolled into C_n, with no impact on the operation of the fuller algorithm. Finally, it is readily composable between nodes, allowing for the function to be well-defined over multiple chains in the task graph. For example, across a given plan over the task in Figure 3.4:

$$C_{SBDG} = C_2 + C_5 + C_8 = p_2c_2 + p_5c_5 + p_8c_8 \tag{3.2}$$

and in general:

$$C_P = \sum_{i \in P} C_i = \sum_{i \in P} p_i c_i \tag{3.3}$$

This presents a very direct means of calculating the effectiveness of a plan, however it does not on its own present a solid means of calculating the expected costs associated with nodes on the graph- it is important for the purposes of planning that we be able to calculate the metric across the graph entirely to avoid the pitfalls associated with the earlier chain-based algorithms.

The expression for the expected cost at any node, then, can be phrased in terms of the expect costs of plans associated with that node, along with the expected cost of the subsequent nodes resulting from those plans. For example, at node C in Figure 3.4:

$$E(C_C) = p_6(c_6 + E(C_E)) + p_7(c_7 + E(C_F)) \tag{3.4}$$

Where the dependent relationship defining hierarchical relationships is explicitly stated above in the inclusion of latter expected costs. This presents a problem for computation, however, given that $E(C_C)$ cannot be calculated itself until $E(C_E)$ and C_F are known. We can ameliorate this complication by noticing that not all subtasks possess a future dependent cost- nodes D, E, and F in our example are all immediately prior to the goal state which, if it possesses a cost, is clearly not dependent on any latter step. We can thus write out the expected costs from these nodes without dependent terms:

$$\begin{aligned} E(C_D) &= p_8c_8 \\ E(C_E) &= p_9c_9 \\ E(C_F) &= p_{10}c_{10} \end{aligned} \tag{3.5}$$

With this information in hand, we can then readily complete the expression for $E(C_C)$:

$$E(C_C) = p_6(c_6 + p_9 c_9) + p_7(c_7 + p_{10} c_{10}) \tag{3.6}$$

,

which handily illustrates how we may iteratively determine the expected cost across the entire graph, from G to S, incorporating all paths by implementing a back-to-front calculaiton system of tiers. Figure 3.5 shows the delineation of tiers across our example graph, where the tier identification itself is noted to be a function of maximal link distance from the goal state. I.E., all nodes a given distance from G will occupy the same tier- it is possible that any node my have links from multiple tiers, but for the need of computing expected cost, only the number of steps needed to calculate all its dependencies matters, and so the links between tiers need not be as clean as in our example, as we will see in our experimental case.

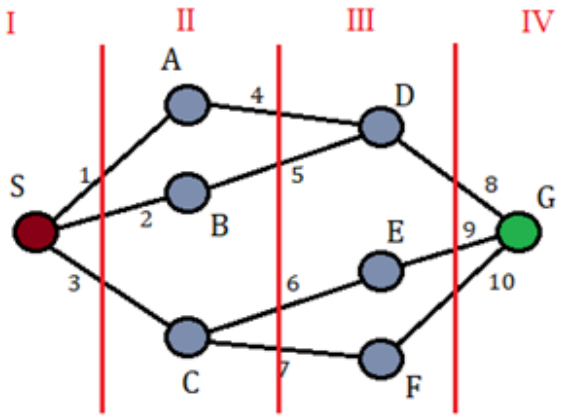

Figure 3.5: Division of the task graph into depth from goal tiers, illustrating the levels of expected cost dependency

3.3 Expected Cost Models

Identifying the configuration of manual and autonomous subtask modules which allows for the optimal cost path through T_G is a difficult problem. A brute-force approach

would be to evaluate the least-cost path from S to G for every possible autonomy level plan. However, this would immediately require evaluating 2^K variants of T_G, which is computationally intractable. In our prior work, [79], we experimentally determined that human operators intuitively select the autonomy level associated with the lowest level of error that would not increase execution time. This indicates that human optimization of autonomy level plans tended towards a Pareto-Optimal solution, which allows us to approach the problem from a different angle.

We create a joint cost function for the expected task performance as calculated at the starting state. We evaluate whether each module should be executed manually or autonomously on basis of the effect doing so has on the cost as seen from the starting state. This approach allows us to focus on the impact on total cost for the entire task, and it relieves us of the need to calculate costs across all possible paths from S to G.

To construct this joint cost function, we fist specify a model for combining the cost of a single subtask into an expected cost. In this model, grouped transitions between subtasks represent outcomes that are related in probability. Figure 3.9 shows the set of transitions leading from the example subtask N to either M or P. For such a transition we take two measures, probability of the event and expected cost of the result. For instance, we have a probability p_{NM} of the transition from N to M occurring, and the cost $\mu(N)$ associated with it as well. The grouped nature of these transitions accommodates the possibility that M and P, as results of executing subtask N, may be statistically independent from the outcome Q as a result of specific actions taken in the course of executing N.

Furthermore, we include a mechanism for incorporating subtask failures during execution. Each **failure mode** occuring at module N has an associated **failure cost** $C_{FNX..Z}$, distinct from μ, which represents the expected loss associated with failure to correctly execute module N and transition the system to states $X..Z$. As an example, if execution time is the cost of choice, the time required to reset the system to the initial state, independent of any subtask, could be a such failure cost. Also, the associated probability of failure for transitions from node N is $1 - \Sigma_{\forall y \in X..Z} p_{Ny}$.

The relationship between separate grouped transitions in T_G, using expected cost, is also highlighted in Figure 3.9. In this example, we have the grouped probabilities for transitions into two groups. The first group consists of transitions from N to M, N to P, as the failure to successfully progress from N to either of these. Alternatively, the other group contains only the N to Q transition. By allowing for such groupings, we account for the existence of multiple choices in the execution of a subtask, which may have differing outcomes without resorting to the inclusion of multiple copies of each node in T_G with disparate outcomes.

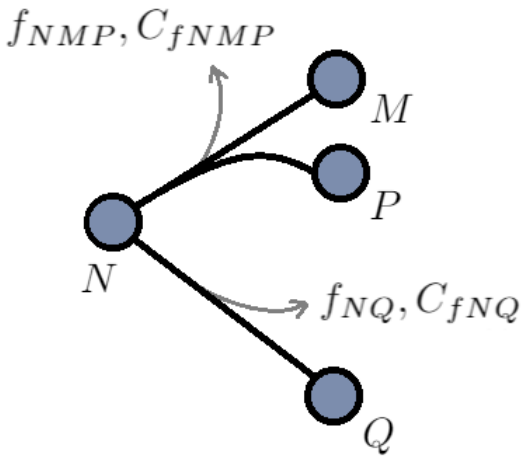

Figure 3.6: Model of expected cost failure grouping on a hypothetical node N, with a joint probability of transitions and failures to M and P, and a singular transition/failure group to Q, illustrating how related subtask transitions are grouped (as in Eq. 12), as well as unrelated transitions which proceed from the same subtask node

Using this model, we can construct an alternative expected costs of subtask execution. While $\mu(N)$ represents the cost associated with a specific subtask, the expected cost for a group $\{N, X, ..Z\}$, denoted as $E(N|N..Z)$, will include not only $\mu(N)$, but also a combination of C_{Nx}, which are the expected costs for the subsequent subtasks (x) which may result from N, as well as the failure cost for the group $C_{F_{NX..Z}}$:

$$(E(N)|NX..Z) = \tag{3.7}$$

$$\mu(N) + \sum_{\forall x \in X..Z} p_{Nx} E(x) + (1 - \sum_{\forall y \in X..Z} p_{Ny}) C_{F_{NX..Z}} \tag{3.8}$$

In which $(E(N)|NX..Z)$ is the expected cost at N given the grouped probability set $\{N, X, ..Z\}$, with the p_{Nx} representing transition probabilities from N to subsequent subtasks in this group. $E(x)$ is the expected cost of subtask x in $\{X, ..Z\}$ as calculated for succeeding nodes in T_G.

This expression allows us to calculated the expected costs associated with each different grouped probability set which may be present for a subtask N. To calculate the joint cost associated with the subtask N itself, we can utilize the minimum such calculated expectation out of all subtask groups resulting from N:

$$E(N) = min(\{E(N|NX_1..Z_1), E(N|NX_2..Z_2), ...\}) \tag{3.9}$$

The model herein necessitates that the expected cost at each subtask is dependent on all potential future subtasks. We will be using the impact on the expected cost at the start state as our objective function, however it introduces the complication of recursive dependency into the calculation of that overall expected cost.

We can resolve this complication by working backwards in T_G from the goal. G itself has no subsequent subtasks and is the terminal state, so its expected cost can be considered zero. All subtasks immediately preceding G therefore depend only on their own transition probability and cost, allowing calculation to proceed from back to front until reaching S. T_G can thus be divided into **subtask tiers**, defined as their minimum path length to S, as illustrated in Figure 3.7.

Using this recursive definition, we can then readily calculate $E(S)$ as a result of different autonomy level plans. Recall that we assigned the lowest autonomy level, $A_L = 0$, as having all subtasks which can be performed manually. Further increases in autonomy level are effected by changing one subtask at a time from manual to autonomous execution. By computing $E(S)$ we can measure the impact of these changes by comparing the expected cost before and after the change.

The calculation in the prior section indicated the means of joining expected cost under the assumption of joint probability- that is to say, for a given set of probabilities for edges from a node N, $\{p_{i1}, p_{i2}, ...\}$ the sum of probabilities adds to one, as all outcomes are

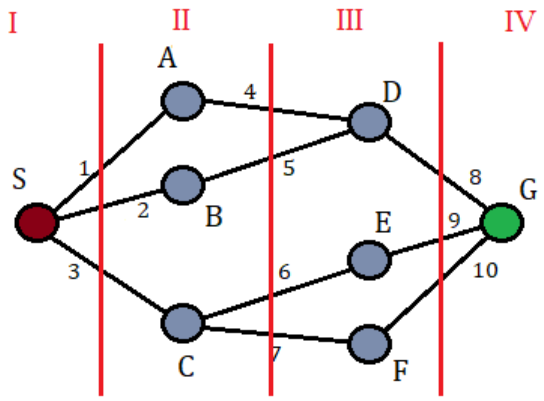

Figure 3.7: Illustration of division of T_G into tiers, by which the progression of expected cost calculations may proceed from G backwards to S, with subtasks on tier IV calculated first, followed by III, II, and finally S in I.

contingent on one another. However, this is not always the case and poses two limitations. First, it does not allow for there to be multiple different joint outcome sets associated with any node, which would stipulate expanding the graph to include a process node for each decision, in spite of the new nodes all bearing the same manual and autonomous code. Second, there is no built-in means for expressing failure modes. One might also implement an additional set of process nodes represent failure states, but doing so by creating paths to the start state would complicate the acyclic structure of the plan graph and cause inconsistencies in cost evaluation.

Instead of attempting to modify the graph in such a way as to preserve these properties, we alter our expectation calculations to include terms for failure modes and select between different grouped path options based on the cost optimization principle of selecting the optimal expected cost. Because the expected cost is a calculated average over the whole graph, we can presume that the planning algorithm's action will express the selection over the optimal path, should it ever make a decision for a given node, and thus calculation over that path is representative.

We present here three models for accommodating error and managing joint probabilities, each of which is an augmentation of the fundamental calculation based on the principle above.

3.3.1 Model I

The first model presumes that each individual edge possesses a singular probability set, with a transition probability and a failure probability. In this application, any transition either succeeds or fails, and only has output states of a failure mode or a single success state, with the joint expected cost selected as:

$$E(N) = min(\{(E(M)|f_{NM}), (E(P)|f_{NP}), (E(Q)|f_{NQ})\}) \tag{3.10}$$

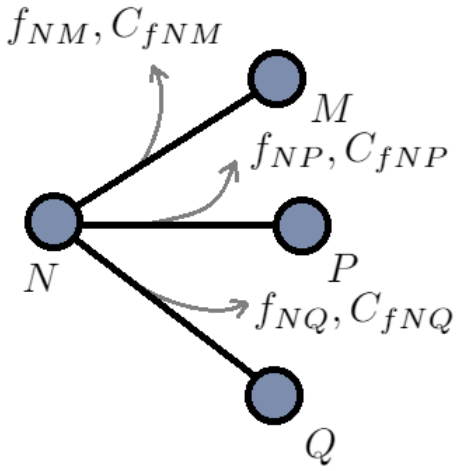

Figure 3.8: Model I expected cost failure grouping

Where for any given node, the conditional expectation $(E(X)|f_{NX})$ is taken to mean the expected cost at X, given the rate and cost of failure specifically associated with the $N \rightarrow X$ transition, and is given by:

$$(E(X)|f_{NX}) = p_{NX}E(X) + (1 - p_{NX})c_{fNX} \tag{3.11}$$

With dependencies in cost resolved identically to the tier-based method outlined in the prior section.

3.3.2 Model II

Model II (Figure 3.9) is a more flexible case, in which some of the edges are represented in joint conditional likelihood sets, and some are singular. This version also requires a decision mechanism, as in Model I, but includes a new mechanism for calculating the joint probability over a grouped set of transitions:

$$E(N) = min(\{(E(MP)|f_{NMP}), (E(Q)|f_{NQ})\}) \tag{3.12}$$

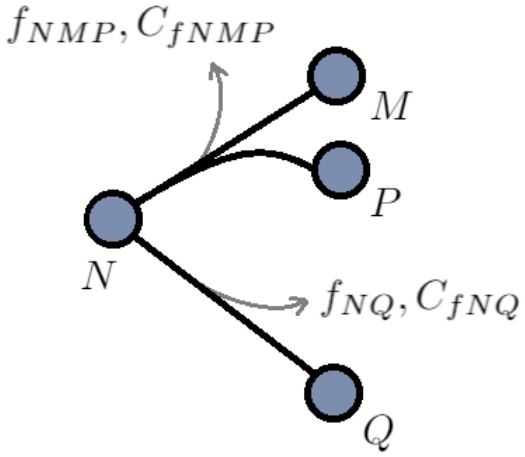

Figure 3.9: Model II expected cost failure grouping

Where the singular model, $(E(X)|f_{NX})$ is identical to the Model I formulation, but the grouped mode is given by:

$$(E(XY)|f_{NXY}) = p_{NX}E(X) + p_{NY}E(Y) + (1 - p_{NX} - p_{NY})c_{fNXY} \tag{3.13}$$

For the example shown, and in the more general case of many edges:

$$(E(X..Z)|f_{NX..Z}) = \sum_{\forall Y \in X..Z} p_{NY}E(Y) + (1 - \sum_{\forall Y \in X..Z} p_{NX})c_{fNX..Z} \tag{3.14}$$

3.3.3 Model III

Our final model, shown in Figure 3.10 is very nearly the same as assumed in the initial discussion of expected cost, with only the additional inclusion of a failure mode and cost function joined with the normal operation cost functions. As this model does not incorporate more than one joint probability calculation, there is no selection mechanism, only a composition function:

$$E(N) = p_{NM}E(M) + p_{NP}E(P) + p_{NQ}E(Q) + (1 - p_{NM} - p_{NP}) - p_{NQ})c_{fN} \tag{3.15}$$

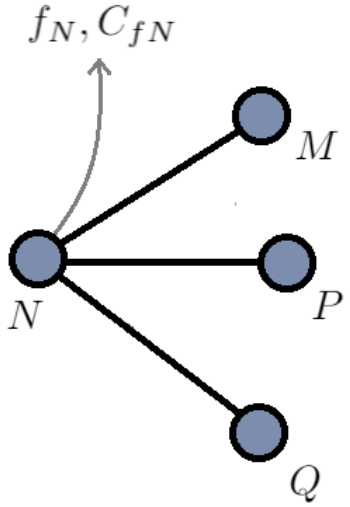

Figure 3.10: Model III expected cost failure grouping

This is the simplest and most tractable model, and so as a general rule we presume to be working with this version unless otherwise specified, however it it worth nothing that aside from the extra step of the decision among minimum costs of available paths, the properties of the joint cost function compositions are identical to this model after selection, as the joint probability objects are all composed expected cost systems.

3.4 Greedy Autonomy Level Assignment

If $E(S_{A_L})$ represents the expected cost at S, given an autonomy level plan for level A_L, the cost change to a higher autonomy level $A_L + 1$ can be written as:

$$\Delta E(S)_{A_L} = E(S_{A_L+1}) - E(S_{A_L}) \tag{3.16}$$

If we calculate $\Delta E(S)$ for each subtask which is a candidate for a transition from manual to autonomous execution (meaning the subtasks which, in the current autonomy level are manual, but may be performed autonomously), we can then identify which subtask P_i will have the *most negative* change in cost when automated, and update the next autonomous level plan accordingly:

$$\rho_{A_{L+1_i}} = A_i, min\Delta E(S)_{A_L}. \tag{3.17}$$

$$\rho_{A_{L+1_i}}[i] = A|min\Delta E(S)_{A_L} \tag{3.18}$$

The change in cost $min\Delta E(S)_{A_L}$ should be negative, as we seek to find an optimal autonomy plan, however, it is possible that some point, all remaining changes induce an increase in expected cost. This point is naturally the local minimum in $\Delta E(S)$, and we define it to be the optimum autonomy level. The resulting VAP+ is described in detail in Algorithm 1.

An illustration of a step of this iterative algorithm is shown in Figure 3.12. In this example, we suppose that the examination for construction of $A_L = 1$ determines that subtask D has the most-decreasing effect on the expected cost at S when converted from manual to autonomous execution, such that we would have $\rho_1 = \{M, M, M, A, M, M\}$. In the second step, we say that perhaps subtask B, with cost changed due to the replacement at D, now has the most improving cost shift and is changed to autonomous execution, for $\rho_2 = \{M, A, M, A, M, M\}$. The algorithm proceeds in the same manner until all autonomy level plans have been constructed, noting the plan for which minimum cost is achieved.

Lemma 1: *The costs of autonomy level plans produced by Algorithm 1 are a convex function of* A_L

Proof: By the update policy, Equation (4), we iteratively change the most cost-decreasing module from manual to autonomous. From Equation (1), we can see that, for

Algorithm 3.1 Iterative VAP+ Algorithm

$\rho' \leftarrow \rho$
while $\mu(\rho) \geq 0$ **do**
  $E(S) \leftarrow \sum_{\forall Y \in \{P[i]\}} p_Y E(Y)$
  $\Delta S_{min} \leftarrow Null$
  $\Delta_i \leftarrow -1$
  for $\forall M_j$ in T_G **do**
    if $M_j \notin \rho$ **then**
      $\Delta\mu \leftarrow E(S) - \sum_{\forall Y \subset M_j} p_Y E(Y)$
      $\Delta\mu \leftarrow \Delta\mu - (1 - \sum_{\forall Y \subset M_j} p_Y) c_{FY}$
      if $\Delta\mu) \Delta S_{min}$ **then**
        $\Delta S_{min} \leftarrow \Delta\mu$
        $\Delta_i \leftarrow j$
      end if
    end if
  end for
  $\rho'[\Delta_i] \leftarrow M_j$
  $\rho \leftarrow \rho'$
end while
return ρ'

a given P_i: $E(P_i)_{A_L+1} - E(P_i)_{A_L} = \mu(A_i) - \mu(M_i)$. Note that all successive subtasks' contributions are unaffected by this change. Further, let us define the set $\{p_{i,1}, p_{i,2}...\}$ to be all transition probabilities associated with paths from S to P_i. By propagating the change in cost for P_i up T_G to S, we can see that $\Delta E(S) = [\mu(A_i) - \mu(M_i)]\Pi_{\forall p_{i,j}} p_{i,j}$. Therefor, for any change, the $\Delta E(S)$ depends only on the probabilities of transitions on paths between S and P_i and the difference in cost between A_i and M_i. Because the probabilities are considered fixed relationships throughout a single run of Algorithm 1, and because **Condition 3** ensures that changing the cost associated with P_i does not alter the cost of any other subtask, $\Delta E(S)$ must be strictly increasing. As such, if $\Delta E(S)$ is initially negative, then $E(S)$ will decrease initially, followed by increase. If $\Delta E(S)$ is initially positive, $E(S)$ will solely increase. In either case, the result is that $E(S)$ is convex.

With this cost function and recursive selection method, we can immediately see that we will, at each step, select the maximally cost-reducing module for inclusion in the new autonomy level plan. Consequently, this algorithm will construct a sequence of autonomy level plans for which the objective function, $\Delta E(S)$, is convex, and therefor the minimum

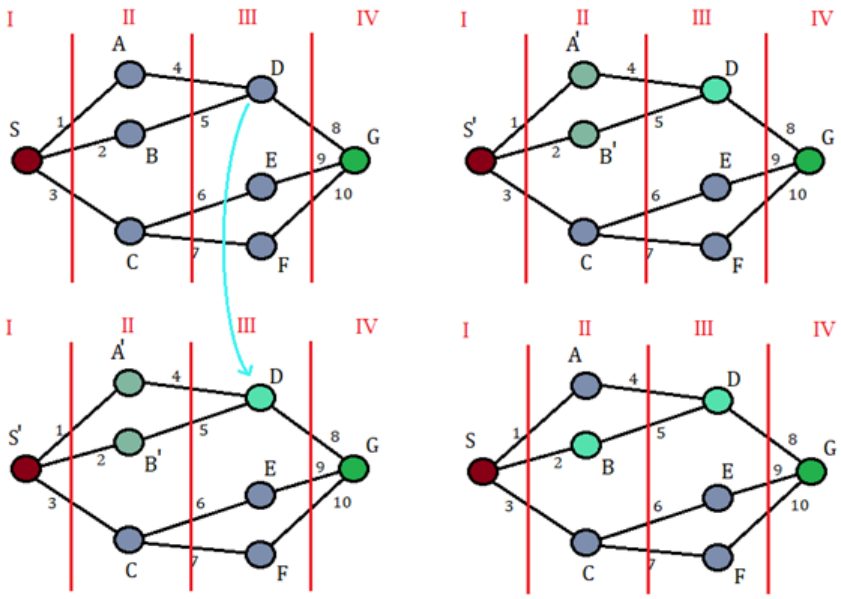

Figure 3.11: Hypothetical Autonomous-to-manual module shift embedded within the task graph, highlighting replacements made on basis of the joint cost function and the cost changes at nodes impacted by the replacement after one iteration of the algorithm, with highlighted green cells represented changes in expectation cost due to module use changes

cost autonomy level plan thus derived will be a local minimum. In the next section we will identify the conditions under which it will also be a global minima.

With this joint cost function in hand, we can then begin to look at the process of defining levels of autonomy based on it. Drawing from the experience gained from the VAP experiments, we want to select a policy which gives the best extension of the principle of selecting the best in class cost optimization, or the lowest cost choice among the most likely paths in T. For the prior experiments, the class argument was regularized around the binary success/failure method rate, but in this case we are optimizing a known objective function and as such need only examine it.

We do need, however a reference frame, which we select to be the expected cost at the starting state. For any given system, the expected starting state cost is a measure of the performance across the full task space and thus presents a key feature lacking in the VAP algorithm of a mechanism for ranking single node modifications in terms of impact on the full planning space. Thus for this algorithm, levels of autonomy will be defined in

terms of node replacements using the metric of

$$\mu(N) = \Delta E(S) = E(S') - E(S) \tag{3.19}$$

Where S' is taken to be the starting state associated with a modified planning graph.

To apply this principle, we can start by noting that any change to a node only propagates changes in cost to preceding nodes, and only if it is, either prior to the change or due to the change, the model-selected choice for representative expectation. So for Models I and II, many potential changes may have no impact on the cost associated with S', though all Model III changes will have some impact, though it may be small after dilution by probability calculations. Our natural step, then, is to examine the change in $E(S)$ due to replacement of an autonomous node process with a manual one. We can then propagate the change in cost back to S, and calculate the effect on the expected cost over the planning graph.

For instance, in our example, the process for examining the replacement of D proceeds as:

$$E(D) \rightarrow p_{HD}c_{HD} + E(DG) \tag{3.20}$$

$$E(A') \rightarrow E(D) - p_{AD}c_{AD} + p_{HD}c_{HD} \tag{3.21}$$

$$E(B') \rightarrow E(D) - p_{BD}c_{BD} + p_{HD}c_{HD} \tag{3.22}$$

$$E(S'_D) \rightarrow p_a E(A') + p_2 E(B') + p_3 E(C) \tag{3.23}$$

Such that:

$$\mu(D') = E(S'_D) - E(S) \tag{3.24}$$

Calculating this cost over all nodes gives a set of cost changes on S, from which the most optimal improvement can be selected, and with which the first replacement of an autonomous node is made by taking:

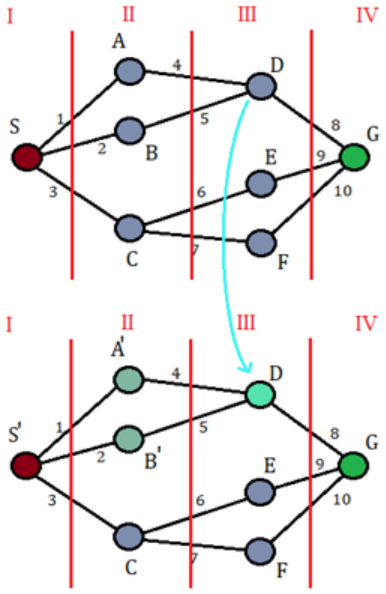

Figure 3.12: First hypothetical Autonomous-to-manual module shift embedded within the task graph

$$X \rightarrow X' | min(\{E(S'_X) - E(S) \forall X\}) \tag{3.25}$$

as illustrated in Figure 3.12 for the case where we assume node D satisfies this condition

This first replacement represents the creation of the first autonomy level for this system, and proceeding from this point, we can take the altered task space, T', and apply the same principle, checking over all toggles between autonomous and manual modes to identify the next most significant gain in $E(S)$, illustrated in Figure 3.13 for the second step, if node B is the next best replacement.

From here, we proceed until no exchanges result in improved cost, thus generating a series of incrementally improved states defining a series of autonomy levels.

This is the means by which we define levels of autonomy under the VAP+ algorithm, but we additionally have a secondary tremendous benefit in that this definition structure comes from proceeding until there are no further increases in efficacy, meaning that the final autonomy level thus defined represent a local peak in the neighborhood of module

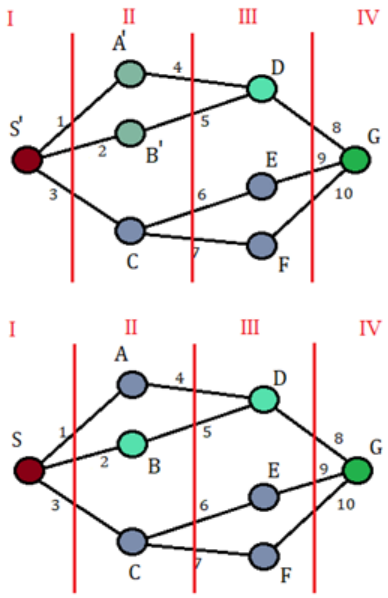

Figure 3.13: Illustration of subsequent autonomous-to-manual module shifts as the algorithm proceeds to identify new autonomy levels

replacements, and thus inherently represents a Pareto optimal autonomy level selection for the system in keeping with out design principle. We could proceed past this set of changes by adopting the least negative impact choice, which would subsequently determine the shape of the cross-sectional efficacy curve beyond the optimum point.

3.4.1 Demonstration of optimizability in A_L

We must also, however, demonstrate two other factors to establish efficacy of this ranking system: first, that there is a statistical coupling between machine performance and human performance which allows the sliding scale to represent the cross-sectional optimum of autonomy. Second, that the flattening of the planning space to this particular ordering leads to a minimization of uncertainty errors in ordering.

For the issue of the coupling between human and machine factors we have a significant leg up in the fact that we have sorted machine modules by their cost function. Because of the monotonic decrease in machine performance inherent in this ordering, we need not examine metric functions in terms pairs of functions, but the measure performance of the manual modes as a function of the corresponding autonomous mode:

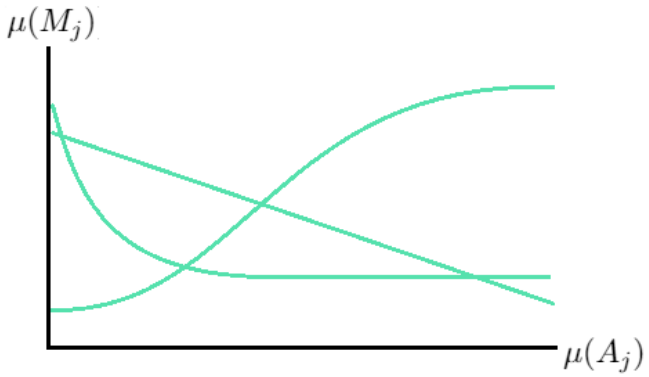

Figure 3.14: Manual performance as a function of autonomous performance

In the most general sense, then, we can define an ordering function which allows us to write coherent functions without regarding the order of the tasks as the independent variable. From this frame of reference we are expressing the cost function for the manual modes in terms of the corresponding index for the ordered set of autonomous mode indices:

$$f_H(j) = \{\mu(M_j)\}|\mu(A_j) < \mu(A_{j+a})\forall j \tag{3.26}$$

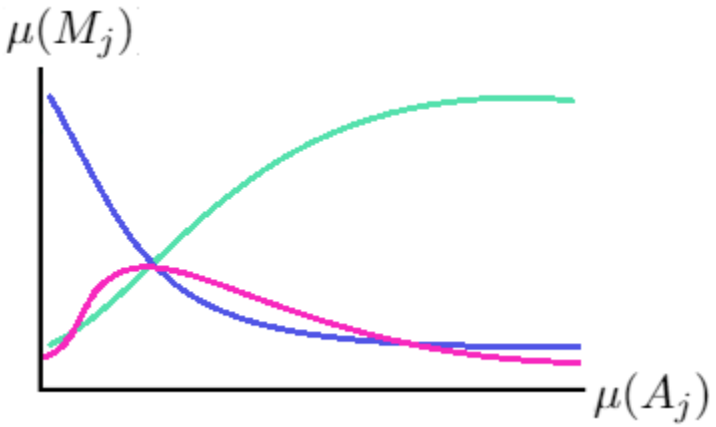

Figure 3.15: Illustration of a hypothetical region of cross-sectional optimality in which machine performance is monotone decreasing (in blue) and manual performance is increasing (in green), with net performance as a function of A_L superimposed on the chart, illustrating how complementary relative performance leads to local optima, as discussed

Where f_H represents the arbitrary relation between the autonomous and manual metric values.

To evaluate the performance of the selection method over these metric functions, we can further define a difference function between a pair of modes:

$$D(j) = \mu(A_j) - \mu(M_j) \tag{3.27}$$

$$D(j) = \mu(A_j) - f_H(j) \tag{3.28}$$

In the prior section, we utilized the joint expected cost at the system starting state as an objective function to be optimized in terms of the autonomy levels. A natural result of the construction of the objective function in this way being that the lowest cost ρ thus identified was a local minimum expected cost autonomy level plan for T_G. In this section, we will analyze this algorithm to identify certain conditions under which this local optimum is also a global optimum. We will also use the analysis to identify the degree of deviation from these conditions under which the VAP+ algorithm still produce globally optimal plans.

For the purpose of clarity, we construct a permutation $i \rightarrow j$ which re-indexes the subtasks such that they are labeled in the ordering generated by Algorithm 1: a subtask P_i is set to use A_i at autonomy level j, in the new ordering it will be labeled P_j.

Using this ordering, we can write out a proxy function for the cost of a plan:

$$\mu(\rho_{A_L}) = (\mu(A_1) \cdot \mu(A_2) \cdot ...\mu(A_{A_L})) \tag{3.29}$$

$$\cdot (\mu(M_{A_L+1}) \cdot \mu(M_{A_L+2}) \cdot ...\mu(M_K)) \tag{3.30}$$

Because of conditions 1 through 3, it is possible to see that this function will possess the same differential landscape as $\Delta E(S)$, the impact of changing the autonomy level on the cost of the autonomy level plan is directly related to the difference in cost between the manual and autonomous modules being exchanged.

Because the change in cost is predicated on the relative efficacy within each subtask, the function representing the relationship between the autonomous module cost and the manual module cost is very useful. We defined this function of manual vs. au-

tonomous module cost, f_H as the relationship, and can thus approximate it by plotting points $(\mu(A_j), \mu(M_j))$ for all subtasks:

$$\mu(M_j) \ = f_H(\mu(A_j)) \tag{3.31}$$

It is important to note that f_H, here, is a continuous approximation of the inherently discrete relationship between $\mu(A_i)$ and $\mu(M_i)$. We can further define an uncertainty function $\delta\mu(M_i)$, to represent variance in the costs of the manual modules relative to the autonomous modules, accounting for the span of measurements across specific subtasks for differing *human users*. Under this assumption, we are presuming that variance among human operators is the dominant factor causing variance across autonomy level plan cost. We can then combine this uncertainty with f_H to calculate a statistical model for the cost over $\mu_\rho(A_L)$, as a function of A_L

$$ln(\mu_\rho(A_L)) = \sum_{j=1}^{A_L} ln(\mu(A_j)) + \tag{3.32}$$

$$\int_{A_L+1}^{K} \delta\mu(M_j) \sum_{j=A_L+1}^{K} ln(f_H(\mu(A_j))) \tag{3.33}$$

This allows us to use the derivative operator $\frac{d}{dA_L}$ to examine cost-change behavior in terms of A_L. Applying the operator to both sides and re-arranging the terms:

This allows us to use the derivative operator $\frac{d}{dA_L}$ to examine cost-change behavior of μ_p in terms of A_L. Applying the operator to both sides and re-arranging the terms:

$$\frac{\mu'_\rho}{\mu_\rho} = \frac{\mu'(A_{A_L})}{\mu(A_{A_L})} - ln(f_H(\mu(A_L))) \int_{A_L+1}^{K} \delta\mu(M_{A_L}) dj \tag{3.34}$$

where $\mu'(A_{A_L})$ is the derivative of the cost function μ evaluated at the autonomous subtask A_L. Seeking extrema, we can set $\mu'_\rho = 0$ to obtain:

$$ln(f_H(\mu(A_L))) \int_{A_L+1}^{K} \delta\mu(M_{A_L}) dj = \frac{\mu'(A_{A_L})}{\mu(A_{A_L})} \tag{3.35}$$

$$\mu(A_{A_L})ln(f_H(\mu(A_L)))\int_{A_L+1}^{K}\delta\mu(M_{A_L})dj = \mu'(A_{A_L}) \tag{3.36}$$

Equation (3.36) constitutes an implicit relationship describing the rate of change of the cost of an autonomy level plan as a function of A_L.

Lemma 2: *There exists one optimal autonomy level plan when f_H and μ' are strictly monotonic functions.*

Proof: Because of Conditions 1 and 2 in Section II(b), μ_ρ is strictly positive, we can note that if an extrema is not at $A_L = 1$ or $A_L = K$ (the boundary conditions), that $\mu'_\rho = 0$ conditionally applies to this extrema. Noting that $\delta\mu(M_{A_L})$ will be a constant, if f_H and μ' are strictly monotone, equation (3.36 will have at most one solution. Thus, the cost of the plan μ_{rho} will have a single extrema, the optimal A_L. This minimal autonomy level plan cost occurs at A_L for which the product of costs for the autonomous and manual modules are balanced against the level of uncertainty in the corresponding manual module. □

Finally, we can combine the results of this analysis to demonstrate that the VAP+ algorithm determines optimal autonomy level plans:

Theorem 1: *When f_H and μ' are monotone, them VAP+ Algorithm 1 produces the globally optimal autonomy levels for a given task graph T_G.*

Proof: By Lemma 1, we know that at least one *minima* of cost exists, the local minima identified by the VAP+ algorithm. In the case that f_H and μ' are monotone, we can assert by Lemma 2 that at most one extrema exists in $E(S)$. Therefor, this singular extrema exists, and hence the local minima identified by the VAP+ algorithm is also a global minima, and is the optimal cost autonomy level plan for T_G. □

However, what this shows is that the net effect of changing A_L depends intimately on the exact form of $f_H(j)$, and further that the pattern of gains due to raising or lowering A_L is not guaranteed without an extant pattern in it. This is of little use when predicting system performance, but is critical for our analysis because it presents a key relationship we can use in system design, notably that f_H appears in this expression in reciprocal form.

Though f_H may appear in many forms shaping the derivative in terms of A_L which can lead to a non-boundary optima, a specific class of functions can be guaranteed to present this- those f_H which are monotonic increasing. This expression is therefor the rigorous expression of the earlier claim that cross-sectional optima exist and are related to autonomy level: If f_H is monotonic increasing, then by definition of the j index parameter by monotonic decreasing $\mu(A_j)$, $\mu(M_J)$ will be inversely related, and thus a cross-sectional optima is achievable by adjustment of A_L.

This relationship, by necessity, establishes that use of j indexed members of A_i as the basis ordering for autonomy level is efficacious in cases in which $\mu(M_j)$ and $\mu(A_j)$ are inversely related. What is most important about this analysis is that it strictly decouples the planning portion of the algorithm from the human performance function, relating the effect of A_L to the derivative of the unknown human function positively under the necessary conditions. It entirely couples the proof of valid optimization under sliding-scale A_L to the sorting function of $\mu(A_j)$, which, presuming that the human user will be using their capability to *select* A_L, fully obviates the need of a relationship between A_L and f_H as the optimization does not need to proceed analytically, but only to present a sliding scale on which the optimization is possible.

Because of this, the goal of allowing variable autonomy as a response to the uncertainty of environmental impacts on performance is very neatly met- changes in the human performance curve, when parametrized by metric sorting, allows the effects of changing conditions to be folded into $\delta\mu(M_j)$, under which paradigm the accommodation of environmental variance is entirely handled by human perception.

3.4.2 Deviations from Monotonicity

From the analysis in the prior section, we can positively assert that problems in which an inverse, monotonic relation between manual and autonomous performance exist are subject to optimization through the adjustment of A_L. However, there are very many possible relations between A_j and M_j which do not fit this guarantee condition, and even

some which are resilient- for instance, in cases where A_j and M_j are directly related, for instance, or those in which the two metrics are simply unrelated. In such cases, the question of applying variable autonomy are vacuous- lacking an optima which is related to the performance of the manual modes, or when the pattern of changes is the same, the optima will not be strongly correlated to the autonomy level, but rather a function of the combinatorial space- i.e., relationships may not be preserved across neighbourhoods, and therefor are not predictable within a scale space constructed from only one metric scale.

However, it is also possible to imagine relationships in which f_H is not precisely monotone, nor precisely inversely related to $\mu(A_j)$, but is close enough to this target that the expression for iterative changes in A_L still yield a local optima. A way we can approach these is by examination of the effects of perturbation of the function f_H. While the shape of f_H is, as a general rule, a feature of the problem space, the parametrization in j allows us some control authority over it. Not precisely by altering j itself which is strictly an ordering index, but by design on the relative positioning of $\mu(M_j)$ by artificially inflating the error rates of A_j to shape f_H.

To see how this can be productive, consider a case with an f_H as seen in 3.16.

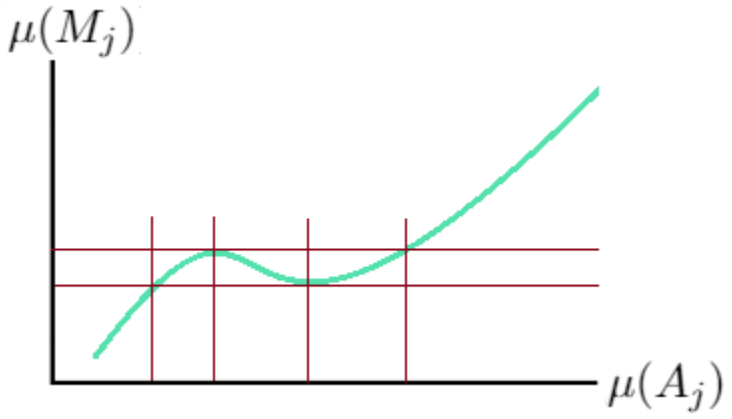

Figure 3.16: An example f_H which is nearly, but not quite, in agreement with the cross-sectional optimality condition, with the offending regions bounded by the red bars

For this kind of curve, it is clear that the deviation from a function which strictly meets the conditions derived above is small, but whether it represents a deviation significant enough to misclassify an optimal point is uncertain. We can model the effect of the

perturbation off a reference curve which does meet the requirements, however, and derive a hypothetical bounding on the error measure by applying a corrective permutation and then comparing this to the uncertainty level needed to effect that change from incidental error.

As mentioned, we can affect changes in f_H by inducing artificial error into $\mu(M_j)$. From a practical standpoint, this amounts to beginning with the known error rate and then further increasing it. For instance, given M_j with $\mu(M_j)$, should we wish to shift M_j by what amounts to a 15% change, we would programmatically modify M_j by 85% of its natural value. If working in probability, we would introduce a random 15% chance of executing a known failure mode, or if time, we would introduce a delay of $0.15\mu(M_j)$, for instance.

We discuss only increases in error as these are the only practical means of shifting the scale j- to insist upon improving the modules would be the alternative, which may not be practical, and certainly not to a level of finesse which would enable a design parameter to be adjusted.

With this adjustment method in mind, what we then need to do is find the minimum shift in $\mu(M_j)$ necessary to eliminate the non-monotonicity. For a given local minimum, this means lowering the corresponding $\mu(M_j)$ to the lowest such level. This recasts the function into an envelope function f'_H, as shown below in 3.17:

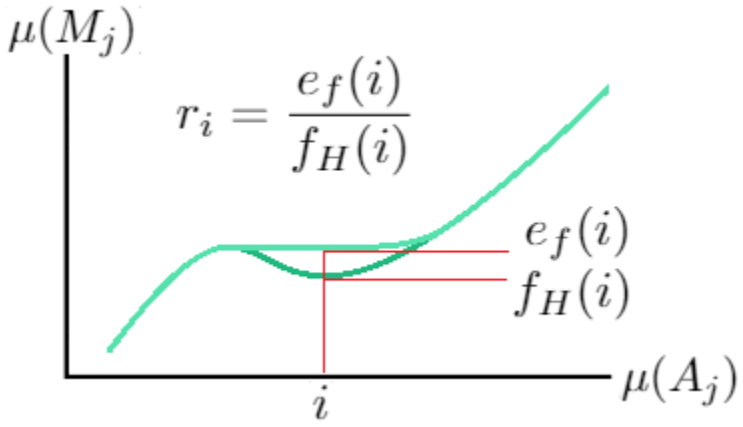

Figure 3.17: An envelope function for f_H, superimposed on the original function

Under this envelope function, we have a valid fit for f_H for application to the above

optimization conditions. Because the optimization condition was predicated on the reciprocal of f'_H, regions under the curve are functional minimizations on the bounding region, which is a convenient knock-on effect of the control authority being derived from reductions of $\mu(M_j)$ rather than shifts in $\mu(A_j)$.

However we have introduced a new source of loss in $\mu(M_j)$- effectively, given the range of reductions needed to bring the curve into compliance, changes in gains due to A_j remain bounded in the positive, but are offset by a given deliberate loss in M_j optimizations- specifically in that reducing the measure in $\mu(M_j)$ makes a shifted autonomous module more likely to be used at any given A_L. This observation is the fundamental downside of the under-curve change to M_j.

Analytically, given a reduction r to $\mu(M_j)$, such that $f'_H(j) = f_H(r \cdot \mu(M_j))$, and the net loss is given by $(1-r) \cdot \mu(M_j)$. From this, we recognize that if the losses due to the expectation of reduced quality of the autonomous modules is less than the expected loss due to selecting the incorrect A_L in the improperly shaped case, then the shift is justifiable.

To calculate the expected change in $\mu(P)$, we can return to the definition for the joint probability. Presuming the adjustment r is being made to module j_r:

$$\mu'(P) = \prod_{i=0}^{j_r-1} p(A_{ji}) \cdot \prod_{i=j_r+1}^{A_L} p(A_i) \cdot \prod_{i=A_L+1}^{K} p(M_i) \cdot r\mu(A_{jr}) \quad |j_r \leq A_L \tag{3.37}$$

For cases where j_r is in the manual modes, then the effect is subsumed because the machine module thus effected does not impact the system, but if the autonomous module impacted is included in the plan, the net impact can be quite easily given by:

$$\mu'(P) = r \cdot \mu(P) \tag{3.38}$$

$$\Delta\mu = (1-r) \cdot \mu(P) \tag{3.39}$$

In a general case, if we presume that an optima exists at $\frac{d\mu(P)}{dA_L} = 0$, then the expected loss in efficacy due to a potential shift can be written by:

$$< \Delta\mu) \approx (1 - r) \cdot \mu(P) \int_0^{A_L} \delta\mu(M_i) \tag{3.40}$$

From which we can use the earlier expression to write, at least, a bounding function:

$$E(\Delta\mu) = (1 - r) \cdot \mu(P) \cdot \mu(P) \prod_{i=0}^{A_L} \mu(M_i) \tag{3.41}$$

$$E(\Delta\mu) = (1 - r) \cdot \mu(P)^2 \prod_{i=0}^{A_L} \mu(M_i) \tag{3.42}$$

And so can write the loss condition as:

$$(1 - r) \cdot \mu(P)^2 \prod_{i=0}^{A_L} \mu(M_i)^{-1} < r \cdot \mu(P) \tag{3.43}$$

$$\frac{1 - r}{r} < \frac{\prod_{i=0}^{A_L} \mu(M_i)}{\mu(P)} \tag{3.44}$$

Specifically, the change r is tolerable if the relative proportion of the change ratio to the shift is less than the ratio of the joint cost of the autonomous modules over the entire cost of the plan. This makes sense, as the expected cost's shift relative to the full span of the cost plans, will necessarily be proportionally scaled to the span of the costs of all the modules. Given that manual modules are unaffected by a shift of an autonomous module, the ratio of the autonomous component cost to the total cost is the relevant scale in which changes can be compared.

As a further case, multiple subtask violations may be resolved- identifying the optimally minimal sequence of shifts which achieves a compliant curve is a complex combinatorial problem itself, but presuming a set of shifts $\{r_1, r_2, ...r_s\}$, the repeating the above procedure give the result:

$$\prod_{i=1}^{s} \frac{1 - r_i}{r_i} < \left(\frac{\prod_{i=0}^{A_L} \mu(M_i)}{\mu(P)} \right)^s \tag{3.45}$$

which is not the most tidy expression, but establishes that there is a threshold limit for certain close curves with multiple small infractions of the conditional cross-section scaling condition in the prior section, and thus genuinely strict adherence to the monotonicity of f_H is not strictly required for optimization along A_L.

In this chapter, I discussed specific manifestations of the weaknesses implicit in the VAP implementation of variable autonomy, and constructed a task model framework which can be used to ameliorate these issues. I further defined the joint expected cost metric for operation within these task models, and from it derived the extended VAP+ algorithm. Using this system, I derived the conditions under which the VAP+ algorithm can determine optimal assignment of autonomy levels in a system implementation, relating them to the cross-sectional optima discussed in prior chapters. I also derived a bound for deviation from the strict satisfaction of the optimality conditions which still permits optimization under the VAP+ algorithm, to allow for extension to more practical cases.

CHAPTER 4

ARNA ROBOT AND SUBSYSTEMS

In this chapter, we discuss the ARNA robot, a mobile manipulator platform on which the experiments using the VAP+ algorithm are performed. Our original experiments implementing picking using a ReThink Robotics Baxter, the key components used being the two-finger parallel gripper, in-wrist camera, and 7-DOF manipulator arm itself on which they are mounted. Due to ongoing maintenance and support issues associated with the Baxter robot, the new experiments are run on a Kinova Gen-3 arm with an attached Robotiq two-finger parallel gripper.

The software for the task is developed in ROS, using the Kinova Kortex package for motion control of the arm and access to the attached Robotiq gripper, and Kinova Vision package for access to the video streams from the RealSense. Subtask modules are implemented within an overarching state machine, each state containing a procedure for either receiving user input via the UI, or performing autonomous controls. All image processing software runs in a parallel with the state machine, making calculated measurements available in real time, and also providing additional feedback to the user through the Heads Up Display withing the UI. Human operators control the robot from a base station which consists of a monitor for visual feedback and a 2-axis joystick.

Work on the ARNA robot pertaining to system development and application to practical problem cases not including the components herein has been detailed in prior work by my colleagues, in [80], [81], and [82].

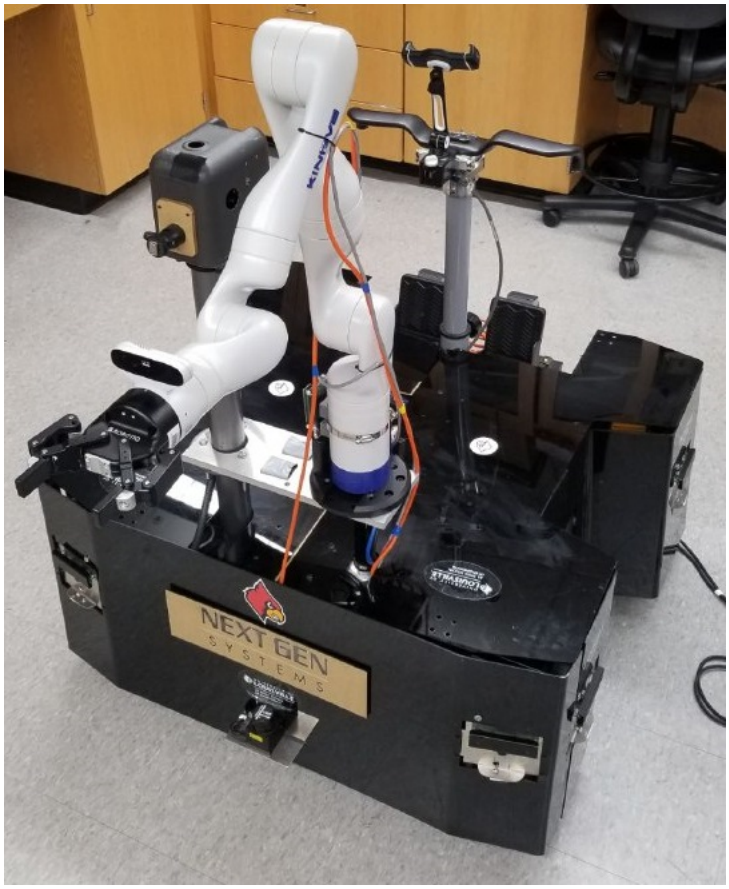

Figure 4.1: The ARNA robot, featuring an omnidirectional base, Kinova 7 DOF manipulator arm, and RGB-D camera system

4.1 ARNA Robot Components

The robot itself is a mobile manipulator, as shown in Figure 4.1, consisting of an omnidirectional base and a Kinova 7-DOF arm. Embedded in this arm is a wrist-mounted Realsense RGB-D camera from which imaging and depth measurements are taken for kinematic profiling and navigation. Control of the system is implemented through an on-board computer, with interface instrumentation via a custom interface board and the EtherCat protocol.

In addition, camera feeds and visual output from some of the subsystems is made available to an operator for assisting with situational awareness during teleoperation control. UI effects include a targeting reticle, angle and position indicators for the end-effector, positioning data from the distance sensors, and an orientation indicator for the base (Figure

4.10).

Operators control the robot via a 2-axis joystick, with the interface allowing selection of different movement profiles of the base and arm, depending on the current subtask under execution. The user may select XY-plane translation of the base, Θ-plane rotation of the base, XY-plane translation of the arm, YZ-plane translation of the arm, and end-effector pitch as the independent motion axis.

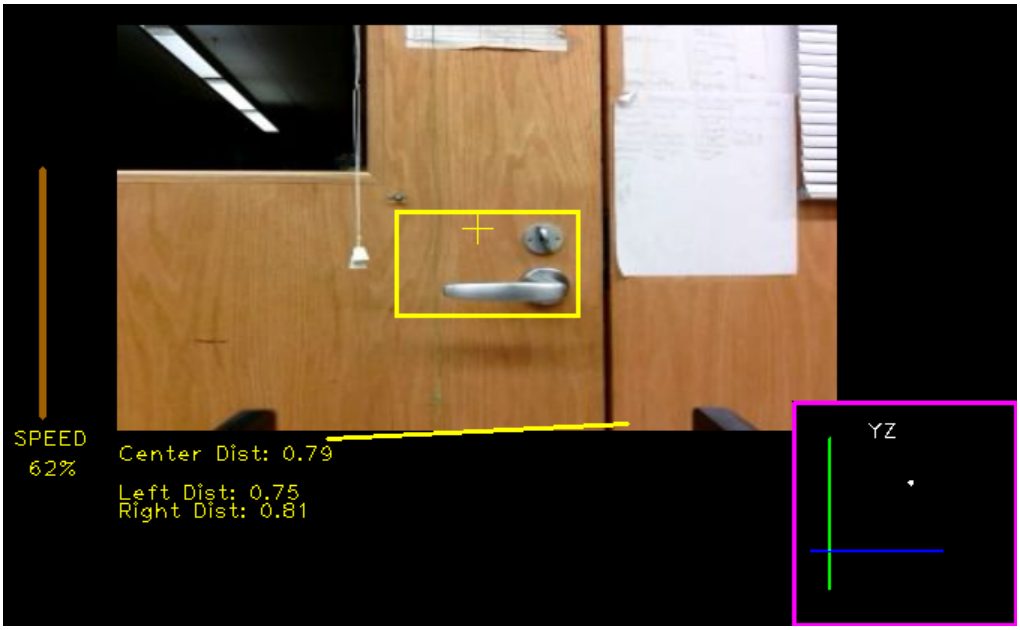

Figure 4.2: Implementation of RGB-D based tracking system for identifying kinematic and navigational goals, showing the speed, orientation, control mode and depth indicators, along with the targeting reticle

The imaging subsystem comprises the software integration for the RGB-D camera and image processing systems that enable tracking of target objects. It operates based on two methods- the use of a Haar cascade classifier to perform single-item target tracking and statistically based depth image analysis system which calculates position of the targets for positioning and tracking. The vision system also supplies the primary feedback for operators in manual modules- output from the measurement systems is displayed as a bounding box around the target, readouts of distance calculations, and a visual indicator of the relative angle of the base to the target.

4.1.1 Bin Picking Vision system

Imaging for this task is provided by the Kinova arm's integrated RealSense camera, with access to the data streams provided by the Kinova Vision ROS package, which publishes ROS topics containing the image data from the color and depth streams. Our software for visual processing subscribes to these topics, processing the images to locate items, identify their positions and produce an angle orientation for later use in the grasping component.

The visual processing system for the original experiment did not include depth imaging, as does the RealSense camera, but rather estimated distance from the end effector to the target object using an embedded IR sensor. In lieu of this, we utilize the depth measurements to calculate the average distance to the sensor in a region around the center of the image. Though acquired through a different mechanism, we utilize this distance identically, adjusting the height of the end-effector until the measured distance is marginally longer than the 'closed' length of the gripper.

This system provides three core functionalities to the robot:

(1) Localization of objects available for picking;

(2) Measurement of distance from the camera to the imaging plane

(3) Estimation of orientation angle of the target object

Object localization begins with segmentation, which takes the initial color image and identifies regions which are associated with the items. We use two methods to generate segments, each implementing a contrast enhancing transform and then applying the Canny edge detector [33] in each of the red, green, and blue color spaces joined by the logical OR operation, with the final segmentation created by logical NOR of the two tracks' output. We previously validated this segmentation method in another publication related to bin picking, [34].

The first process uses a localized gamma transform to increase contrast, followed by a filter convolution which estimates the density of local contours prior to application of the edge detector. The latter process uses an unsharp mask filter to highlight object edges, and proceeds directly to the edge detectors. 4.3 showcases the sequence of operations which

perform segmentation on the color image.

With the Baxter system, distance to the imaging plane was measured using the in-wrist IR distance sensor. Lacking this device in the Kinova, to identify the distance for purposes of approach to the object we use a histogram based method. Using the depth image, we calculate the distribution of measured distances with a 0.5cm bin width (the effective statistical sensitivity of the depth measurement when filtering for noise). From this distribution, we select the peak count at the least distance from the image sensor as the distance to the image plane.

To estimate the object pose, we implement an heuristic based on contact between the minimum area bounding box for the segmented region of the object in the camera image and the rectilinear bounding box.

Here, we describe the image processing algorithms used to register objects in the RGBD camera's field of view, in order to grasp them. The first stage of image processing is color segmentation with an emphasis on guaranteeing that different objects are segmented out, at the expense of segmenting portions of the same item. Because visual properties like color are inherently insufficient for a cluttered space, we instead utilize the visual component of segmentation to identify all divisible partitions, and then, in the second stage of image processing we use the additional information provided by the depth image to rectify the disjoint components. Our fundamental observation is that, because we wish to grasp an object, detection of contiguous faces in physical space is key to manipulation, however, segmenting a specific item in full is not strictly necessary.

Subsequent to image processing, we also implement a pose estimator which calculates grasp angles in the 2D plane for picking operations on the patches identified in the prior sections.

4.1.2 Color Image segmentation

The color segmentation algorithm consists of two preprocessing stages followed by a pair of parallel traditional segmentation methods. The preprocessing stages have two

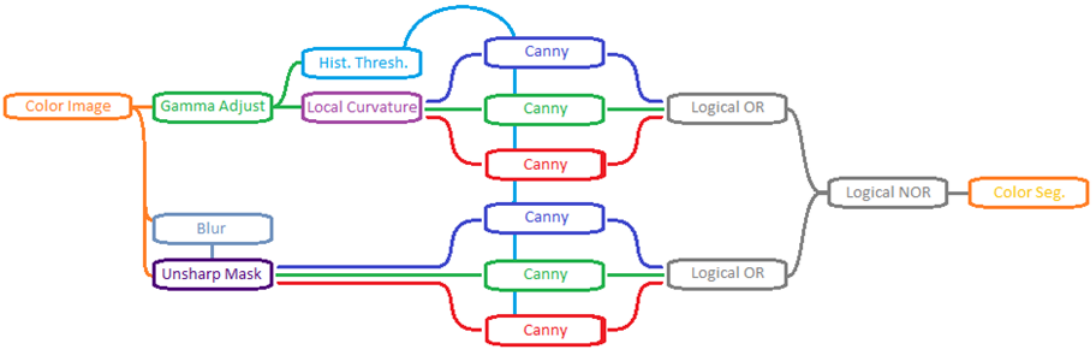

Figure 4.3: Image processing pipeline for used to detect regions in the color image

functions: first, to enhance boundaries due to shadows; and, second, to identify thresholds for edge-detection. Then, two segmentation algorithms are proposed to create overlapping visual partitions. These two algorithms are complementary in the regions in which they are effective, with one generating significant segmentation in visually complex areas, and the other performing more effectively in low-complexity regions.

4.1.2.1 Pre-processing Stage 1: Gamma-adjusted transform

First, we apply a transformation intended to accentuate small visual boundaries between items of similar coloring, based on the traditional gamma-correction transformation in image processing. For a general image, it is defined by the following exponential:

$$V_{out} = AV_{in}^{\gamma}, \tag{4.1}$$

in which V_{out} is the output pixel value, V_{in} the input pixel value, and gamma the correction factor, which traditionally is constant over the whole image.

In our case, to emphasize the contrast shift in certain areas, without washing out others, we instead apply it as a local transformation. We calculate for each pixel an individual exponent by applying multiple blur filters of varying aperture size to a black and white copy of the image. This serves to highlight areas of highly varying contrast, while leaving low contrast regions ignored. If we represent the blur kernel with the operator $K_n(\cdot)$, where n refers to the number of steps of blurring, we may write the new transform as:

$$V_{out} = AV_{in}^{\frac{1}{n}K_n(V_{in})}. \tag{4.2}$$

The recursive definition of the blur kernel K_n in eq. (4.2) is summarized in Algorithm 1.

Algorithm 4.1 Localized Gamma Transform

$C_{blur} \Leftarrow C_{image}$
for i from 0 to n **do**
  $C_{blur} \Leftarrow blur(C_{blur})$
end for
$C_{gamma} \Leftarrow C_{image}^{[1-C_{blur}]}$
return C_{gamma}

4.1.2.2 Pre-processing Stage 2: Histogram thresholds

In the subsequent segmentation algorithms, we use the Canny edge detector to identify region boundaries, which requires a pair of thresholds representing edges, weak edges, or non-edges [18]. These levels are typically free parameters that must be tuned, however, instead we propose histogram thresholding. We determine the upper and lower thresholds by splitting the image intensity histogram about the average pixel value, and then taking the integral average over the two partitions. The pseudo-code for this stage is as follows in Algorithm 2:

Algorithm 4.2 Determination of Upper and Lower thresholds from the image histogram.

Input: C_{gamma}
Output: L_{thresh}, U_{thresh}
$lower\_sum \Leftarrow \frac{1}{2}\sum_0^{avg_x} hist(C_{gamma})$
$L_{thresh} \Leftarrow argsum_{lower\_sum}\sum_0^x hist(C_{gamma})$
$upper\_sum \Leftarrow \frac{1}{2}\sum_{avg_x}^{max_x} hist(C_{gamma})$
$U_{thresh} \Leftarrow argsum_{upper\_sum}\sum_0^x hist(C_{gamma})$

Algorithm 2 provides a metric which is designed to pick values that evenly divide the intensity spectrum of the image in the upper and lower halves of the distribution, which approximates the locations of the quartiles. In this way, the former free parameters are connected to image properties, rather than requiring tuning which would subject the

algorithm to environmental variance.

4.1.2.3 Processing Stage 1: Local Complexity

After pre-processing, we use a measure of the local curvature as a complexity scaling metric by which edges are selected. To estimate this measure, we first apply the Canny edge detector to a black and white copy of the image, and then convolve a uniform density kernel over the image. This acts to sum over the convolution window the number of pixels which contain edge marks in the neighborhood of the center pixel. This metric represents the arc length of curves near the central pixel, the sum total representing 'complex' and 'simple' cells. The pseudo-code for this stage is as follows:

Algorithm 4.3 Processing Stage 1: Local Complexity.

Input: $C_{gamma}, L_{thresh}, U_{thresh}$
Output: C_{seg_1}
$C_{contours} \Leftarrow CannyEdge(C_{gamma}, L_{thresh}, U_{thresh})$
$C_{comp} \Leftarrow convolve(C_{contours})$
for red, green, blue **do**
  $C_{color} \Leftarrow CannyEdge(C_{comp}^{color}, L_{thresh}, U_{thresh})$
end for
$C_{seg\_1} \Leftarrow C_{red} \vee C_{green} \vee C_{blue}$

This segmentation method, especially in conjunction with the adjusted gamma pre-processing, creates images in which areas with multiple contrast boundaries become highly segmented. This allows us to divide portions of the image which contain high-frequency information. By contrast this method tends to perform poorly on highly smooth low-frequency areas, especially those with weak color gradiation.

4.1.2.4 Processing Stage 2: Pseudo-unsharp Mask

The second segmentation method applied is based on the sharpness-enhancing transformation of the unsharp mask, used to reduce perceived blur in an image, and equivalent to a high-frequency range amplification. In the first segmentation pass, we focused on selecting for high frequency areas, so the implementation of a high pass filter may seem counter-intuitive, but the idea is to amplify the high-frequency, low-amplitude features

present in smooth areas to detectable levels. The pseudo-code for this stage is as follows:

Algorithm 4.4 Processing Stage 2: Pseudo-unsharp Mask.

Input: $C_{gamma}, L_{thresh}, U_{thresh}$
Output: C_{seg_2}
$scale \Leftarrow \frac{L_{thresh}}{U_{thresh}}$
$C_{filter} \Leftarrow C_{gamma} - scale \cdot blur(C_{gamma})$
$C_{unsharp} \Leftarrow C_{filter} \cdot C_{gamma}$
for red, green, blue **do**
  $C_{color} \Leftarrow CannyEdge(C^{color}_{unsharp}, L_{thresh}, U_{thresh})$
end for
$C_{seg\_2} \Leftarrow C_{red} \vee C_{green} \vee C_{blue}$

Because unsharp masking acts by subtracting a smoothed image from the original, this means that rather than acting as a filter that enhances both smooth and complex areas, the complex areas tend to become highly noisy- as such this phase is effective primarily in smooth areas.

The next portion of the color segmentation is to join the pair of segmented images and identify their region properties. As with the color channel conjoin operation, we simply use the logical OR. The purpose of this operation is to leverage the mutually preferable qualities of each in the same output segmentation. Finally, calculation of centers of mass, areas, and bounding boxes of the object surface patches thus identified is performed by the application of the marching squares algorithm to the regions defined by the segment boundaries. The complete color segmentation algorithm pseudo-code is summarized in Algorithm 4.

We illustrate the contrasting results of processing stages 1 and 2 in Figure 4.4, which was obtained by processing three images from sample databases [16][17]. It can be noted that the areas in which the two segmentation stages perform effective division is mostly disjoint, but in total is evenly distributed over the image space. Additionally, there is a certain amount of overlap, but between the two images, we can see that the conditions in which each algorithm dominates alternates.

4.1.3 Depth Segmentation

In addition to color segmentation, We also segment the depth image, so as to use regions within the distance profile as the correction metric when joining disparate color segment patches. The depth image is naturally well-conditioned for the purpose of identifying the useful physical faces needed for grasping.

Depth segmentation begins with utilizing the unsharp mask, which is useful for amplifying the small contrast between physically close, but distinct faces. We then utilize the Canny edge detector once more to locate region boundaries, with image-derived thresholds. For the depth thresholds, we use the lower and upper quartile of the unsharp mask filter, since the unsharp filter is a rough measure of the distribution of local gradients. Finally, we define the identified regions as the non-boundary areas of the image, and use the marching squares algorithm to label the blobs defined by these areas. The resulting depth segmentation pseudo-code is summarized in Algorithm 5.

Algorithm 4.5 Depth Segmentation Algorithm.

Input: D_{image}
Output: D_{seg}
$D_{filter} \Leftarrow D_{image} - blur(D_{image})$
$D_{unsharp} \Leftarrow D_{filter} \cdot D_{image}$
$min_D \Leftarrow min(D_{unsharp})$
$max_D \Leftarrow max(D_{unsharp})$
$L_{thresh} \Leftarrow min_D + \frac{1}{4} \cdot (max_D - minD)$
$U_{thresh} \Leftarrow max_D - \frac{1}{4} \cdot (max_D - minD)$
$D_{contour} \Leftarrow CannyEdge(D_{unsharp}, L_{thresh}, U_{thresh})$
$D_{seg} \Leftarrow MarchingSquares(1 - D_{contour})$

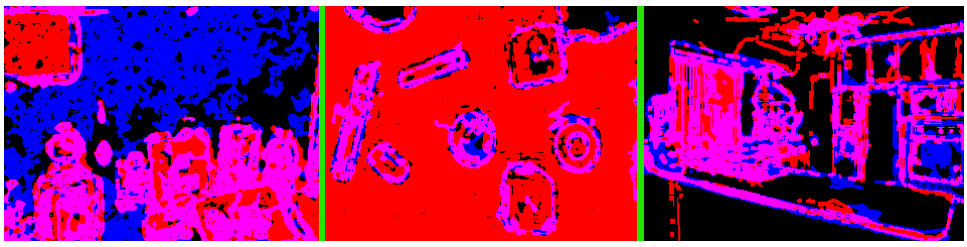

Figure 4.4: Comparisons of regions of effect between processing stages 1 and 2, highlighting the difference in coverage among three images taken from sample sets.

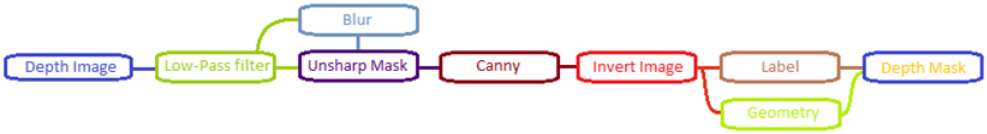

Figure 4.5: Image processing pipeline for used to detect regions in the depth image

4.1.4 Color and Depth Fusion

Figure 4.6: Method for joining regions in the depth and color spaces into coherent graspable surfaces

Once the color and depth images have been segmented, we proceed to recombine the color image patches via the depth image. We correlate the depth and color image planes by the known geometric offsets between the two cameras.

For each patch, P, with bounding box B_P we define a neighborhood $N(P) = (1 + \frac{|B_P|}{|C_{image}|})$. We mark as adjacent any other patch P' which meets the following criteria:

1) $B_{P'} \cap N(P) \neq \{\}$; and

2) $mode(C_{seg}[P] \cap D_{seg}[P]) = mode(C_{seg}[P'] \cap D_{seg}[P'])$

Concurrent to constructing this adjacency graph, we utilize Kruskal's algorithm[19] to label connected segments by presuming edge weights of 1 between adjacent patches and 0 for non-adjacent pairs.

4.1.5 Grasp estimation

Once object regions are identified, we utilize their area properties to estimate a grasp pose, guided by the force-closure principle [7]. Under force closure, a grasp is optimized when an arbitrary force may be exerted on the object through the contact of the gripping member. In a 2-dimensional, 2-finger gripping normal to the image plane, a good estimation for this grip is along the line through the center of mass of the item which has minimal length between contact points.

Because the physics of items is unknown or unmodeled, we approximate the item center of mass through visual image processing. To determine the grasp orientation, we simply utilize the smallest edge of its corresponding minimal bounding box. The corresponding grasp pose calculated as a rotation around the image Z axis as illustrated with a bottle bounding box shown in Figure 4. In this example, the bounding box in blue was identified from image patches, while the red box is aligned with the XY image coordinate frame.

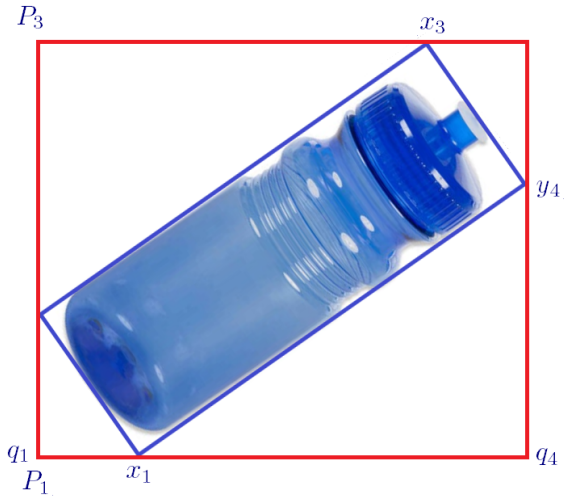

Figure 4.7: XY Cartesian bounding box contact with the object bounding box.

Examining the geometry illustrated in Figure 4.7 allows us to calculate the relative angle between these boxes by utilizing the known contact locations x_1, x_3, y_2, and y_4, as well as the XY box corner coordinates:

$$w = \sqrt{(x_1 - P_1)^2 + (q_4 - y_2)^2}, \tag{4.3}$$

$$h = \sqrt{(x_3 - P_1)^2 + (q_4 - y_4)^2}, \tag{4.4}$$

where w and h are the width and height of the minimal bounding box, respectively. We may then calculate the areas of the two boxes, A_{bb} the area of the XY rectangle, and A_{ob} that of the minimal rectangle:

$$A_{bb} = (P_3 - P_1)(q_4 - q_1), \tag{4.5}$$

$$A_{ob} = w \cdot h. \tag{4.6}$$

Then by defining the rotation angle of the minimal box from the XY box as θ, from simple geometric analysis, we can observe that:

$$A_{bb} - A_{ob} = hsin(\theta)hcos(\theta) + wsin(\theta)wcos(\theta) \tag{4.7}$$

$$A_{bb} - A_{ob} = (h^2 + w^2)\frac{sin2\theta}{2}. \tag{4.8}$$

From equation (4.8), we can find the grasp rotation angle θ:

$$\theta = \frac{1}{2}sin^{-1}(2\frac{A_{bb} - A_{ob}}{h^2 + w^2}). \tag{4.9}$$

4.1.6 Disinfection Arm & Base Control systems

To successfully sanitize the targets, a coverage planning algorithm based on restricting motion of the actuator along a conic prism region surrounding the target, determined by the measurements from the vision system, is implemented. In the autonomous mode, this algorithm generates a trajectory along the surface defined in 4.8, which allows for control of the 7-DOF manipulator

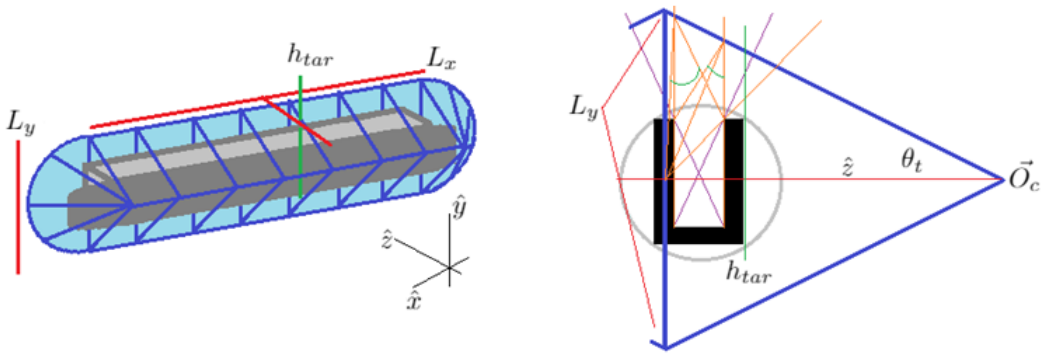

Figure 4.8: Bounded kinematic profile for the surface sanitization tracking components of the ARNA System, illustrating Degree of Freedom reduction for manual control modules

4.1.7 Object Tracking for the Disinfection Task

The imaging subsystem is the software integration to the RGB-D camera and underwrites the localization, tracking, and kinematic bounding submodules. It operates based on

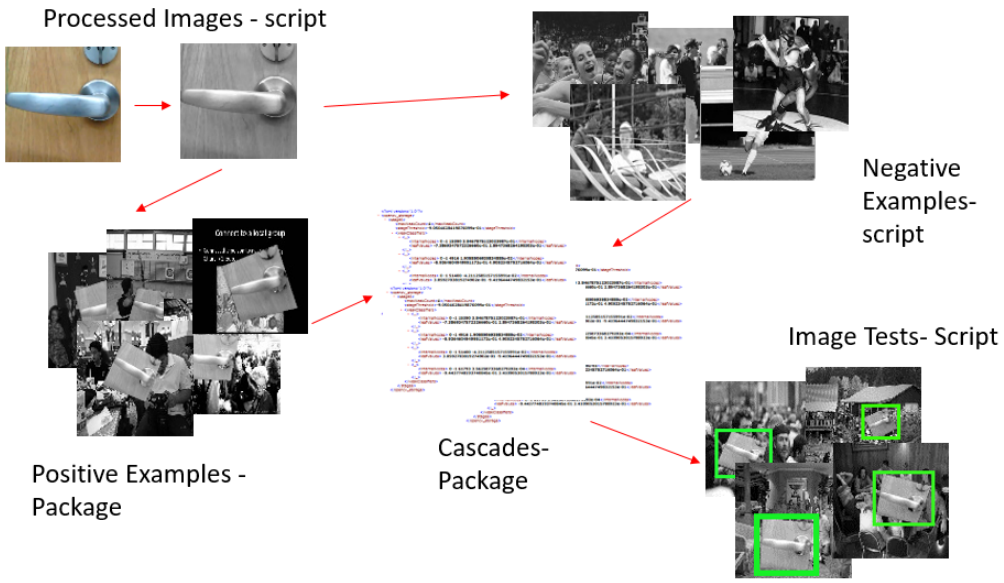

Figure 4.9: Image processing workflow pipeline for generating high-quality single-item Haar Cascade Classifiers for tracking objects in the field of view

two methods- the use of a Haar cascade classifier to perform single-item target tracking and a statistically based depth image analysis system which calculates position of the targets for positioning and tracking. 4.9 highlights the image processing pipeline for generating classifiers for tracking individual items of interest, such as door handles, efficiently- single instance classifiers can be generated using this process.

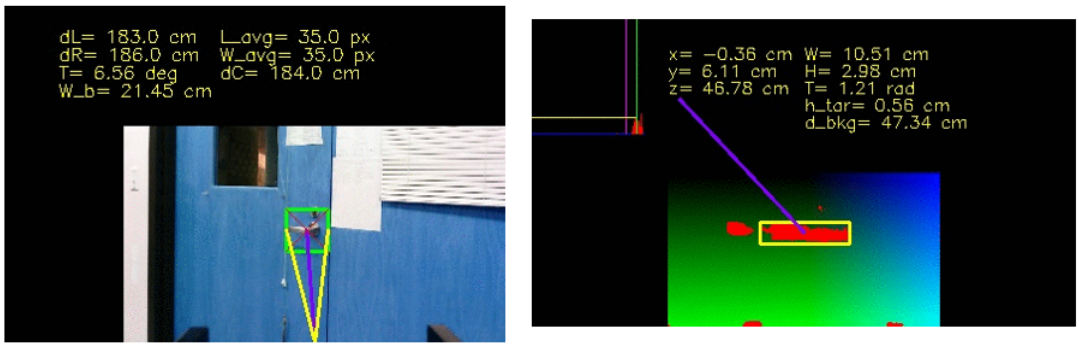

Figure 4.10: Implementation of RGB-D based tracking system for identifying kinematic and navigational goals for the ARNA Sanitization system submodules

4.10 highlights the diagnostic output of the depth and tracking image interfaces, in which the measurements of the object postion, location relative to the camera, and te depth based profile of the target are displayed. These data are the inputs which manage

the functionality of the navigation and coverage subprocesses.

4.1.8 ROSFuse Instrumentation Protocol

To manage the capture of low level sensor data for safety and navigation, we developed a modular software protocol for extending a variable dataspace within a microcontroller firmware system (MCU) that allows robotic sensor data to be streamed via the Robot Operating System (ROS) architecture. This protocol copies the data formatting structure inherent to ROS messages and implements a local DN bridge to allow for asynchronous bi-directional data transport over any communication channel. We implement a demonstration of this system on a mobile robot test bed to manage communications between the sensor data acquisition MCU and the primary control computer, and use this test case to measure the efficacy of the protocol through latency and packet loss, and tracking validation by comparison to other measurement systems on the robot.

Our protocol was designed to possess following advantages over other frameworks:

- Scalability: Our protocol demonstrates strict linear relations between packet length and delay, and provides a mechanism for adding devices by editing only a configuration file.

- Simplicity: our structure is designed specifically to copy the ROS ethos of configuration, allowing changes to variables to be made entirely independent of the source code

- Efficiency: we have made use of the ROS publisher/subscriber model for datasharing through message types. This allows us to implement variable sharing with minimal latency.

In this section, we detail the design of the ROSFuse communication protocol. ROSFuse is structured to topically mimic ROSs publisher/subscriber framework, including the use of message definition files to define data types. The primary goal of the data share bridge is to make available in the ROS namespace variables which are set within the MCU, and vice-versa.

At the top level, this is achieved by a pair of processes: a ROS node running on

the CPU, and an interrupt-driven process on the MCU. The ROS node retains a list of the topics initialized for data sharing, and the MCU process transfers data to and from variables internally. The ROS node subscribes to topics shared with the MCU and publishes topics shared from it, while the firmware interrupt routine parses data into variables, and transmits shared variables. This full workflow is illustrated in Figure 1, and the procedure for message parsing in Algorithm 1.

Algorithm 4.6 ROS_Fuse Parsing Algorithm

```
Function: ROSFuse_Parse(Controls,Peripherals,Port)
if Port.available() then
  while Port.next() ≠ "&" do
    Port.read()
  end while
  packet ← ""
  while packet[−1] ≠ EOL do
    packet ← packet + Port.read()
  end while
  Message ← packet.split(",")
  for ∀peripheral ∈ Peripherals do
    if peripheral.label == Message[0] then
      Current ← peripheral
    end if
  end for
  for datum ∈ peripheral do
    if datum.type[a] == Float then
      floatMembers[a] = Message[a]
    end if
    if datum.type[a] == String then
      stringMembers[a] = Message[a]
    end if
    if datum.type[a] == Int then
      intMembers[a] = Message[a]
    end if
  end for
end if
for control ∈ Controls do
  packet ← "&"
  for datum ∈ control do
    packet ← packet + String(datum)
  end for
  packet ← "EOL"
  Port.write(packet)
end for
```

4.1.9 Packet Structure

In order to implement any communication protocol, we must first select a packet type for the transmission itself. We select a simple string-based, delimited structure, as indicated in Figure 2.

The advantages of using this format are three-fold:

I. Use of a string format simplifies parsing- though a numerical formatting system would enable for lower-overhead communication, the use of message namespace conventions when coupling to ROS dramatically simplifies the process of integration.

II. Messages formatted with start and end characters via the native string type can be of functionally unbounded length

III. The use of the String type obviates the need for a complex encoding protocol for carrying strings- because packets are patched directly into a ROS topic, matching packet names to topics is both convenient for readability and identification.

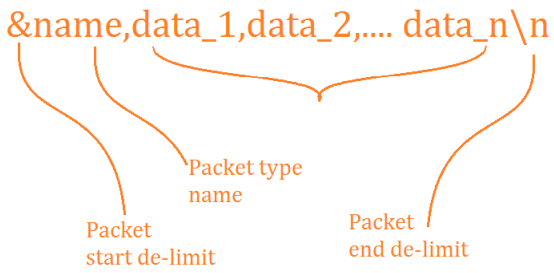

Figure 4.11: Transport-layer packet structure

As a further note, because the use of message type descriptors is copied from ROS to the firmware, there need not be any data included within the messages themselves to guide type selection, reducing packet size.

4.1.10 Message Definitions

In keeping with the design goal of matching the operational characteristics of the data bridge to ROS, we define two generic ROS message types to correspond to transmitted data. Each message type contains variable length arrays to store data, illustrated in Figure

3.

On initialization of the ROS bridge node, the configuration file describing the message is read, generating instances of these topics. This configuration file serves to replace a concrete message definition for each data packet, enabling broad scalability. Further, the parameters for the hardware communication layer are within this configuration file.

Each message contains three primary fields- arrays for containing floating point, string, and integer data. In both the ROS and firmware packages, the data type field determines which array a specific datum is stored in. For instance, if the 4th member of a packet is an integer, then it will be stored in the 4th element of the integer member array, This indexing structure lets the MCU side system store any data type within a single object.

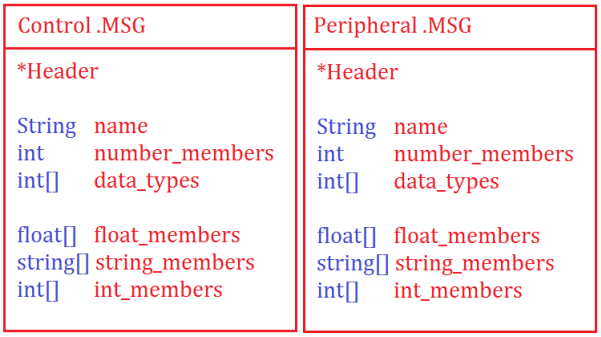

Figure 4.12: ROS messages corresponding to Controls and Peripherals

4.1.11 MCU Firmware

The primary component of the firmware side of the bridge is a C package, which defines the object handlers for both types of message, as well as a utility function for the transmission layer.

Within this package, there is an allocation for master lists of controls and peripherals. For controls, this list provides access to the variable list associated with the transmission, and for the peripherals the list of transmissions to be made. Objects for each message are created within the main execution loop and populate the list at time of creation.

Each peripheral initializer takes as input the transmission channel, name of the packet, number and type of data members associated with the transmission. The construc-

tor for controls follows the same profile, where the process of variable insertion is handled upon receipt of a serial packet. Within the serial parser, a packet label is searched in the list of packet names for the associated control object, and data is placedinto the objects corresponding data arrays.

The transmission read function required as input only a pointer to the object handler, and therefor is suitable for use within interrupt routines.

4.1.12 CPU Software

For the ROS implementation the task of reading, parsing, and writing to the transmission is augmented by the additional task of building and maintaining the ROS topic space. The ROS node package defines object types for the peripherals and controls, and packet parsing and variable storage components are exact parallels of the used in the MCU, just as the message structure and corresponding objects are analogous.

The ROS component begins with the configuration file, an example of which is illustrated in Figure 4. This file defines the ROS topics and subscribers. It begins a subscriber topic for each control, and a publisher for each peripheral. The primary operation loop of the script alternately reads the transmission queue, collects peripheral reports and publishes them into the ROS topic space.

By contrast, control topics for the MCU are attached to the corresponding subscribers. When one of these topics is published, the subscriber collates the data from the publication, formats it into the transmission format, and writes it to the transmission medium.

4.1.13 Implementation System

We have built a custom interface board based on the PJRC Teensy 3.6 Arduino-compatible microcontroller. This MCU was selected for its high clock rate and large amount of IO, which considerably simplifies the PCB design.

Several of the sensors communicate over multi-device protocols, with 32 total devices

```
# Essential Serial port information
<SERIAL>
  <COM>/dev/ttyACM0
  <BAUD>115200

<CONTROL>
  <DEVICE_NAME>relays
  <NUM_PARAMETERS>3
  <DATA_TYPES>2,2,2

<PERIP>
  <PERIP_NAME>block_1
  <NUM_READINGS>14
  <DATA_TYPES>2,2,2,2,2,2,2,0,0,0,0,0,0,2
  <UNITS>
bool,bool,range,range,range,adc_raw,adc_raw,roll,pitch,yaw,accx,
accy,accz,temperature
```

Figure 4.13: Example of the configuration file for building a set of publishers and subscribers in the ROS bridge node, structured so that each message does not require a custom MSG definition in ROS

reporting a total of 56 measurements. In our application, the sensors are arranged in 4 blocks of 8 each, with the control outputs being routed through individual terminal lines on the board.

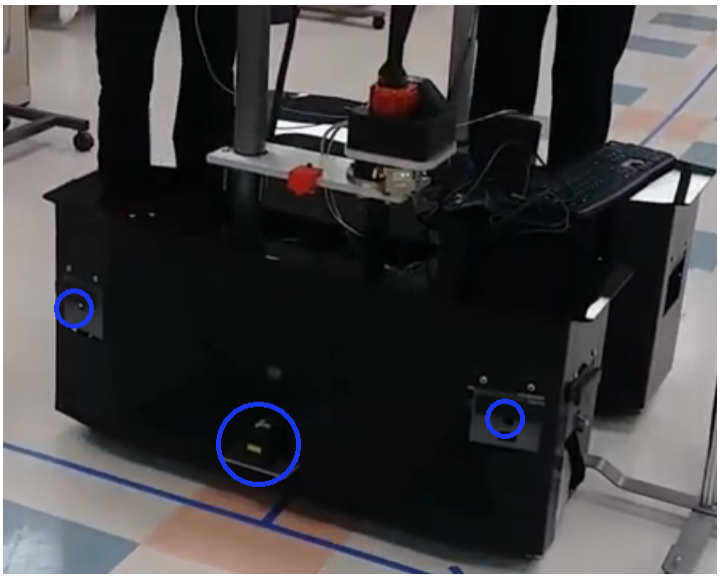

Figure 4.14: Image of the ARNA mobile robot, with the locations of the two front-side sensor blocks and the LIDAR highlighted

4.1.14 Transmission Medium

For this system, we are using the built-in USB/Serial connection for transmission. The Teensy USB/Serial adapter always communicates at the USB limit of 12 Mbit/s, but we enable the standard hardware serial as well, enabling alternate baud rates for testing the effect of transmission speed on queuing loads and data rates.

4.1.15 Sensors

The sensor set was chosen to implement navigational assistance, making for a particularly diverse set of sensors. This is useful for testing as it enables increasing the number of sensors, variation in the data types, and the proportion thereof. As mentioned, the sensors are arranged in four blocks. Each block contains three ultrasonic distance sensors, two button based contact sensors, two IR distance sensors for cliff detection, and one 9-DOF IMU.

4.1.16 Kinematic Controls for Bin Picking

For this task, we are using a 7-DOF arm for positioning of the camera and grasping the item. However, two factors limit the utility of full-rank kinematics.

The first is that the original experiment utilized a 2-axis joystick, with the task divided into subtasks acting in at most 2-dimensions. This was primarily related to the action of the modules running on the Baxter robot- the Baxter's physical repeatability was relatively poor compared to other robots, and communication latency within the reverse kinematics limited the deployment of feedback control. The practical impact of this was that both autonomous and manual modules required short corrective segments, which utilized the imaging frame, confining these adjustments to 2 dimensions. In keeping with the original experiment setup, we have kept the subtask implementation the same.

The second, and more technically substantial reason, is that the tracking algorithm operates in the 2 dimensions of the imaging frame. For both the Baxter, and the Kinova, however, the imaging system's pointing vector is aligned with the principle axis of the

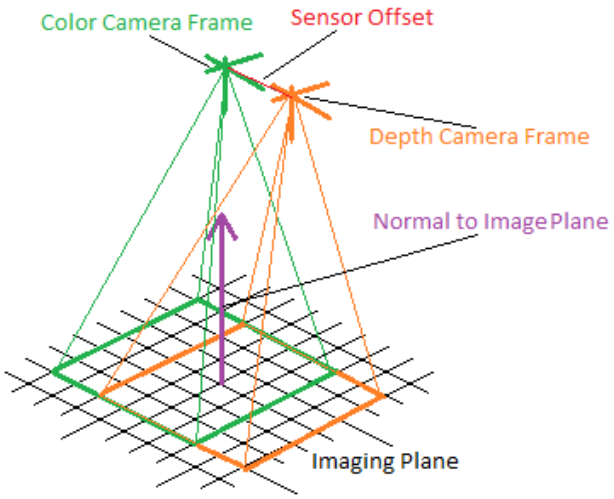

Figure 4.15: Illustration of the geometric relationships between the imaging plane, color and depth image frames, and the gripper pointing vector. The co-linearity of the color camera, depth camera, and gripper means the transformations between the reference frames are constant. Because the trajectory for the arm implements a velocity controller, tracking the target object in the camera frame requires only a static offset transformation to align the gripper in the final step of approach prior to grasping.

gripper. As such, the coordinate transform between the camera frame and the gripper frame is entirely within the image plane.

This means that it convenient and efficient to restrict the picking problem to four total dimensions- X and Y positioning within the camera frame, which axes are perpendicular to the gripper principle axis; the Z-axis perpendicular to the image frame, which can be conceptualized as the distance from the camera to the imaging plane, and finally the roll of the end effector, which acts as a rotation in the image plane. With controls to confine the picking surface to the imaging plane, we can implement controls entirely in this 4 dimensional workspace. This is used to confine the kinematics to the 4 dimensional frame discussed above, identifying the pitch and yaw rotation of the target surface relative to the robot frame (Previously addressed by strenuous calibration of the initial imaging pose using the Baxter IR sensor).

To ensure that the robotic system has sufficiently stability that performance variations are dominated by human behavior and capabilities, we employ a sliding mode con-

troller designed around proportional control regions. Sliding mode controls are well-known to be robust to disturbances [36], making them excellent for visual tracking, as measurement disturbances due to image noise, movement of the camera, and lag from image filtering will not prevent the controller from converging to the target. Our sliding mode controls are constructed by regional tuning of proportional controls, with the switching functions controlling the regimes in which each proportional constant acts. Input signals are measured from the RGB-D imaging system, as described previously. For each of the X, Y, Z, and Θ_Z axis of the arm.

The switching functions are given by:

$$\sigma_{armX}(e_x, e_y, e_z) = 0.05(|e_x| > 5cm) \tag{4.10}$$

$$\sigma_{armY}(e_x, e_y, e_z) = -0.01(|e_y| > 25cm) - 0.05(|e_y| \leq 25cm) \tag{4.11}$$

$$\sigma_{armZ}(e_x, e_y, e_z) = 0.01(|e_z| > 25cm) + 0.05(|e_z| \leq 25cm) \tag{4.12}$$

$$\sigma_{arm\theta_z}(e_x, e_y, e_z) = 10.0^\circ(|e_{\theta_z}| > 15.0^\circ) + 5.0^\circ(|e_{\theta_z}| \leq 15.0^\circ) \tag{4.13}$$

With the final 4 DOF controller for the arm system:

$$< \dot{x}_{arm}, \dot{y}_{arm}, \dot{z}_{arm}, \dot{\theta}_z > \tag{4.14}$$

$$= V_{LIM}[\sigma_{armX} e_x, \sigma_{armY} e_y, \sigma_{armZ}, \sigma_{\theta_z} e_z, e_{\theta_z}]$$

Where V_{LIM} is a peak velocity selected as a safety margin, set at 40 cm/s in our experiments to keep the peak energy of the robot/object system below a 1 Joule injury threshold [37].

4.1.17 Kinematic controls for the Disinfection task

To ensure that the robotic system meets the requirements for the autonomous module uncertainties being low, we employ a sliding mode controller designed around proportional control regions. For the controller, the input signals are measured from the RGB-D

imaging system. Θ is approximated by a proxy δH, the difference in distance between y measurements made at the edges of the calculated bounding box for the tracked object.

For each of the X, Y, and Θ axis of the base, and the X, Y, and Z axis of the arm, switching functions for the controller were determined by tuning a proportional controller for each region of approach independently, and combining the final constants into a common control system.

$$\sigma_x(e_x, e_y) = [0.5(e_y > 60.0cm) + 0.15(e_y \leq 60.0cm)] \tag{4.15}$$

$$\cdot [(0.015(|e_x| > 30.0cm) + 0.01(|e_x| \leq 30.0cm))] \tag{4.16}$$

$$\cdot (|e_y| > 25cm)V_{LIM} \tag{4.17}$$

$$\sigma_y(e_x, e_y) = 0.25(|e_y| > 5.0cm)V_{LIM} \tag{4.18}$$

$$\sigma_\theta(e_{y1}, e_{y2}) = 0.15V_{LIM}(e_{y1} < 25.0cm)(e_{y2} < 25.0cm) \tag{4.19}$$

Where V_{LIM} is a peak velocity selected for safety, and from these functions. The control system for the arm is implemented similarly, albeit more simply than the base, with switching functions:

$$\sigma_{armX}(e_x, e_y, e_z) = 0.05(|e_x| > 5cm) \tag{4.20}$$

$$\sigma_{armY}(e_x, e_y, e_z) = -0.01(|e_y| > 25cm) - 0.05(|e_y| \leq 25cm) \tag{4.21}$$

$$\sigma_{armZ}(e_x, e_y, e_z) = 0.01(|e_z| > 25cm) + 0.05(|e_z| \leq 25cm) \tag{4.22}$$

$$\sigma_{arm\theta_z}(e_x, e_y, e_z) = 10.0^\circ(|e_{\theta_z}| > 15.0^\circ) + 5.0^\circ(|e_{\theta_z}| \leq 15.0^\circ) \tag{4.23}$$

$$[\dot{x}, \dot{y}, \dot{\theta}] = [\sigma_x(e_x, e_y)e_x, \sigma_y(e_x, e_y)e_y, \sigma_\theta(e_{y1}, e_{y2})|e_{y1} - e_{y2}|] \tag{4.24}$$

$$[\dot{x}_{arm}, \dot{y}_{arm}, \dot{z}_{arm}, \dot{\theta}_z] = [\sigma_{armX}e_x, \sigma_{armY}e_y, \sigma_{armZ}e_z, \sigma_{\theta_z}e_{\theta_z}] \tag{4.25}$$

4.1.18 User Interface

Interfacing with and controlling the robot is performed through two systems- direct or SSH access to the embedded computer on the robot, or via a tablet interface. The tablet interface includes manual control systems for both base navigation, and positioning of the robotic manipulator, all connected over ROS topic and service systems, allowing integration over multiple computing systems for ease of access.

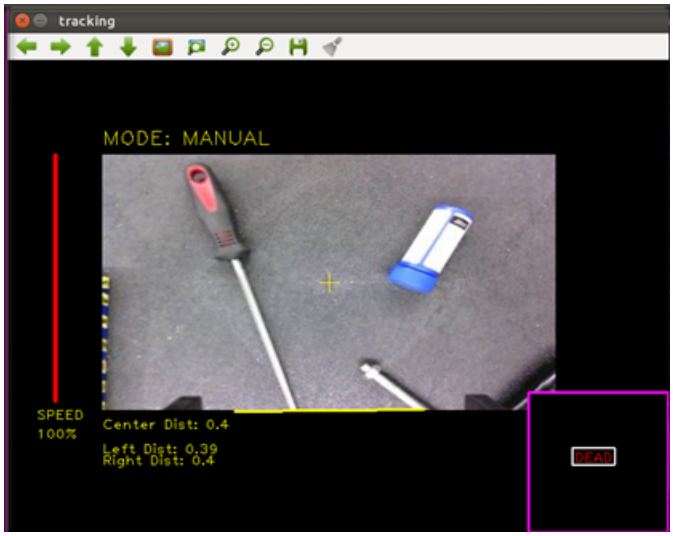

Figure 4.16: Image of the user interface presented to the operator, including prominently the visual feedback camera feed, the the autonomous/manual mode indicator, measurements for the center, left-hand side, and right hand side distances, and the control mode indicator.

In addition, camera feeds and visual output from some of the subsystems is made available to an operator for assisting with teleoperation control situaitonal awareness as well as debugging and experimental evaluation.

To interact with the robot and perform these tasks, we have designed a user interface, including a visual feedback system and a joystick input system. The visual feedback component, seen in Figure 4.16 consists primarily of the color image stream display, from which the user can observe the workspace of the robot. It also includes the system mode indicator, which notes whether the system is currently acting under manual or autonomous operation; distance indicators for the center, left, and right sides of the imaging frame, to assist with depth perception, and a control indicator, in the bottom right, indicating either

Figure 4.17: The base station from which the operator controls the robot during the experiments, showcasing the monitor with UI and the joystick. In these experiments, the user is not permitted to observe the robot workspace directly, only receiving visual feedback through the UI. In the image, the whiteboard to the right blocks the view of the robot.

which phase of autonomous operation, or which set of axis are controlled by the joystick in manual mode. As can be observed in Figure 4.17, the operator receives visual feedback exclusively from the UI camera stream, a screen blocking direct observation of the robot.

When operating manually, the user has access to three different control axes for the robot. The first is movement in the XY directions associated with the imaging plane, controlled by both axes of the joystick. The second is linear motion in the Z direction, elevating or lowering the end effector. The final control is for the rotation in Θ_Z, the roll axis of the end effector.

In addition to this, in the manual mode the user has two buttons which control additional functionality- the first button toggles the axes control mode through the three settings described above, and also a non-acting state in which the arm will be held in a steady state. The second button toggles the gripper between the open and closed state. The user may toggle the gripper at any time, though it is worth noting that only grasping

requires it, and the autonomous grasping module will automatically open the gripper prior to execution of the final approach, regardless of the initial state.

In this chapter, I described the ARNA robot, the platform on which the experiments in the subsequent two chapters are performed. I laid out the description of the robot's form as an omnidirectional manipulator, and the hardware components present as assets for task completion. I also described the combined computer vision, human interface, low-level communications and control systems designed for application to the disinfection and bin-picking tasks performed by the robot.

CHAPTER 5

ROBOTIC DISINFECTION WITH VAP+

To validate the performance and utility of the VAP+ algorithm on a real world problem, we apply it to the problem scheduling intervention planning for a mobile robot manipulator engaged in a sanitization task. This task presents an excellent opportunity for testing as it is highly non-sequential (as discussed in Section I), and has multiple methods for each point of operation in which differing task execution both exists, and has divergent implications on future task accomplishment for the robot.

5.1 Task Description

One of the primary goals of variable autonomy systems, besides identifying a specific conditional ideal autonomy level is the ability to react to environmental changes. In the case for the VAP, this was discussed as being a component of the human's selection of the A_L parameter, which was predicated on the autonomous module performance and thus allowed adaptation to conditions to remain a responsibility of a manually-driven task. In the case of the VAP+ algorithm, however, this mechanism is not available, directly.

We could, hypothetically, implement the level identification algorithm to generate a proxy function for A_L and allow an operator to dynamically set it, shifting between the plans identified for each version of T' corresponding to the selected level, but this fails to make use of the excellent property that the VAP+ algorithm identifies natively the peak performance level of the system. If we are to make use of this property, though, that means that adaptation to environment must take place at some level other than the planning phase, as the execution on the graph is a deterministic procedure.

One arena where the graph can be shifted without changing its structure, however, is

in the expected cost function. While changed the function itself is vacuous- the calculation for C already embeds variant weightings and models- it is possible to shift the known costs and probabilities in response to the environment. In this way, the problem of casting the autonomy level as a function of input data is relaxed to the problem of predicting failure probabilities or costs, which is far more well established in real applications.

If we use methods like this, then we can repeatedly apply the changes in cost or probability to shift the weights, recalculate the expectations across the task space, and then plan with a new level of autonomy, effecting a constrained, well-defined response to the environmental shifts.

To experimentally validate our algorithm and the predictions of our analysis thereof, we have applied it to the optimization of Human/Robot interaction on a physical robot performing a task with several subtasks and multiple possible execution paths.

We collected operational data comprising subtask costs, in terms of time, and the probabilities of successful completion of each subtask for both the robot operating under autonomous execution and for 15 human operators executing manual modules for each subtask. Using these values, we applied Algorithm 1 to identify autonomy level plans, and then collected the performance data for execution of the derived autonomy level plans. Additionally, we use the data collected to evaluate the aforementioned analytic results, namely Equations 3.36 and 3.44.

This validation experiment was implemented on the Adaptive Robotic Nursing Assistant (ARNA) robot performing a disinfection task. The purpose of this task is to render potentially contaminated target surfaces (such as door handles) in a working environment free of viral contaminants by either spraying disinfectant or shining UV light onto the target surface. To accomplish this, the robot must identify a target, approach this target, align its end-effector, and execute a trajectory of the end effector tool. Data collected for this research focuses on the execution of these subtasks from a kinematic perspective, rather than the efficacy of the sanitation process itself.

5.1.1 Experimental Procedure

Our experiment consists of two parts, with three objectives. The first component of the experiment is to measure performance characteristics for human operators and the robot system, as input to the VAP+ algorithm. The latter portion of the experiment collects data under operation using the autonomy level plans determined by the algorithm to validate the planning stage. The objectives of the experiments are to collect the necessary data for developing autonomy level plans under the VAP+ algorithm, to validate the optimizing behavior of the algorithm, and to check the predictions of analysis in Section II(D) and (E), specifically Equations 3.36 and 3.44.

The sanitization task begins with the robot in a random orientation facing towards the target object- a door handle, in this case. The robot must approach the object, align its frame parallel to the surface behind the object (the door supporting the handle), position the end effector with the object, and execute a short trajectory around the object. The course of each trial runs essentially the same, regardless of phase- the user, through the previously discussed UI, executes the manual portion of the tasks, and the robot executes the autonomous tasks. For the first phase, the experiment is either entirely manual, or entirely autonomous, to collect the requisite performance data. In the second phase, execution proceeds based on the generated autonomy level plan.

5.2 Experimental procedures

The experimental process to examine this problem class will proceed much the same as for the VAP algorithm, with a beginning task of acquisition of module-specific success and failure rates, as well as unit costs, being measured. For this task, the primary operational minimizing quantity is time, as none of the modules is resource consumptive to a notable degree- for this system, comestibles are primarily constant rate of operations-based, and thus execution speed and accuracy are the objects of interest.

For each test, then, the full manual and autonomous process versions may be executed, with internal branches selected such as to uniformly sample all transitions rather than

nodes, as nodes are representative of the fixed process, whereas transitions are associated, for the purposes of task planning, with expenditures and failure states.

Below, for each task, we further break the tasks down by failure mode model, selecting the specific mode and the failure-based conditions and costs:

1. Target Identification- Target ID is a process in which a failure can be rectified at a later point, and thus is implemented with Mode III, allowing for a transition to the object tracking state. For this state, failure cost can be measured as the additional time expected from the corrective step.

2. Alignment of robot to target- Alignment is, as with identification, rectifiable with intervention and thus to allow for corrections, Mode III is selected again. Cost is likewise also the feedforward cost.

3. Corrections to object tracking- Object tracking corrections are deterministic, and failure at this step will result in a cascade failure, and so Mode I is most applicable, given that the failure results in a process reset, and failure cost is the difference between expected cost and the cost at this step.

4. Assignment of kinematic bounds- Kinematic bounding is best formalized using Model II, as the autonomous process failure requires a reset, but a manual failure need only fail back to target alignment, necessitating a mixed format error model.

5. Selection of tracking mode- Tracking mode selection is as the object corrections, with future failures contingent on performance, and thus implements Model I.

6. Placement of end-effector- Placement of the manipulator is a mixed class problem, and thus implements Mode II.

7. Tracking of coverage trajectory- Because the coverage algorithm is contingent on placement, but can only proceed to the goal state whether manual or autonomous in execution, all failures must transition to earlier states, making Mode III the most fitting.

The selection of each probability model for the transitions effectively determines the sampling batteries which must be performed, with Modes II and III necessitating experiments assuming each grouping of error states, while Mode I requires only a single experiment

set for all states, naturally.

Not that in this case, the sampling need only cover a uniform distribution of transition cases, rather than evaluate all possible paths, as the decomposition to atomic tasks (under the presumption that performance is unimodal and of low deviation) ensures that task accomplishment is independent up to uncertainty. it is notable here to mention that if the distribution of transition probabilities is not unimodal, then it is likely the task contains a hidden modality suggesting that it should be split into multiple nodes. If the distribution is of a singular peak, however, then our assumptions regarding model uncertainty hold, and the error tolerance bounds are retained.

Once the prior probabilities and costs have been identified, and as a matter of course the distributions of task probabilities have confirmed to be sufficiently independent, we can proceed to experiments on the autonomy level. In the VAP experiments, we examined the problem complexity classes across all autonomy levels to those selected by the operator, and for the VAP+ algorithm we can apply a similar method. In this case, we use the algorithm to generate the full set of autonomy levels, implementing trial cases at each level. This will allow the comparison across the algorithm's full scale of autonomy, comparing the range of outcomes to that selected by the algorithm to establish the local optimality experimentally.

In addition, however, by utilizing the span of available data, we will also be able to examine a large array of predicted costs based on the transition probabilities. This comparison will allow us to evaluate the real-world value of the local optima by comparison to the theoretical ranges of expected costs achievable. To find these bounds, we can iterate over the set of all possible chains in the process graph, which presents a sever tractability problem for planning, but as an offline processing task is achievable.

From this data, refinements to the experiments can be made to underwrite a successive experiment set which can establish the validity of the adaptive nature of the algorithm by introducing deliberate environmental changes at sensitive nodes which have the largest impact on the determined autonomy level and observing the changes in planning efficacy under this alterations.

By examining these sensitive nodes, we will be able to put quantitative measures on the established performance range certainty relationship predicted in Chapter 3. Additionally, observation of the shift in local peak performance autonomy level under changing conditions, and the ability of the algorithm to successfully anticipate these changes when presented with the induced prior probability changes, will demonstrate compatibility with dynamic prediction systems in terms of capacity for implementation as an adaptive planning system.

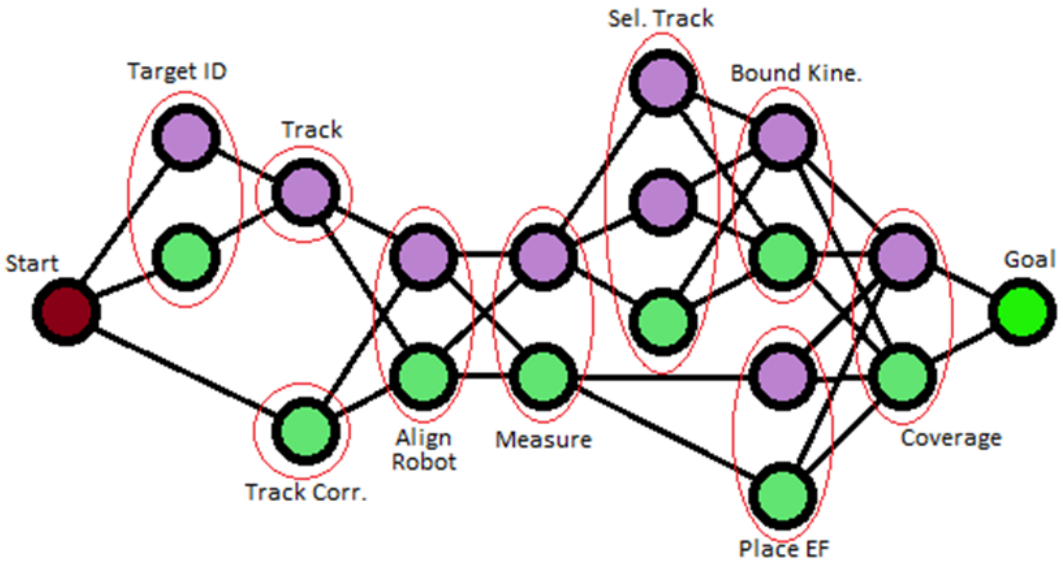

Figure 5.1: Illustration of the ARNA sanitization robot task graph, with groups highlighted, each task group as noted in the text being labeled with its corresponding nodes, and the module labels and indices associated with each subtask module

For the VAP+ algorithm, we decompose our disinfection task into seven sub-tasks:

P_1 : Target Identification- locating the target object in the RGB-D camera's field of view.

P_2 : Alignment of robot to target- maneuvering the robot to be aligned with the object's supporting surface.

P_3 : Corrections to object tracking- Adjusting the position of the robot with respect to the target itself.

P_4 : Assignment of kinematic bounds- Identifying the physical extent of the object and location relative to corrected end-effector placement.

P_5 : Measurement and tracking mode selection- Identifying the calculation model for generation of the coverage trajectory.

P_6 : Placement of end-effector- Adjusting the end effector position and angle to correspond to the correct starting place of the coverage plan.

P_7 : Tracking of coverage trajectory- execution of the trajectory to move the end effector around the object to effect the sanitization protocol.

These represent subtasks P_i in the Section II formulation, with corresponding M_i and A_i illustrated in Figure 5.1, which shows the process graph T_G associated with the subtasks. Each ellipse bounding multiple tasks corresponds to a group of subtasks as in Section II(A).

It is worth noting that not all of these processes need be executed to achieve the task- for instance, manual execution of P_4 can lead directly to P_6, but P_5 is necessary after autonomous execution. Likewise, the target identification stage is unnecessary when the alignment is performed by the operator, but is required for autonomous alignment. In our first phase experiment, paths are chosen to ensure that all modules have sufficient data collected for analysis.

The data are collected throughout each trial by recording the actions of the user via the UI scripting. Time costs are measured as simple duration from beginning one task to completing it, as recorded within the system software. Failure chance is measured in two ways- first, all instances in which the robot system encounters a significant fault, such as a collision, or a tracking excursion in which the target is fully lost, a failure instance is recorded for all modules, under the presumption that the eventual fault may have been precipitated by poor performance of any preceding module. Additionally, for manual executions, cases where the operator attempts to achieve a subtask, but must re-try the task are marked as a failure state for that specific subtask module.

5.3 Experimental Results

We have collected data for transition probabilities and module cost, measured in terms of time, for each of these processes across 15 human operators and the autonomous modules. Users executed each manual module individually under teleoperation, and from these data, relative cost functions are constructed, and we can use Algorithm 1 to generate autonomy level plans for each A_L. Subsequently, the users execute the calculated autonomy level plans to collect performance data for the assignment algorithm. Human subject authorization was granted under IRB 18.0659 to test the adaptive interface.

Table I shows the probabilities and execution times for each of the subtasks as performed for both the autonomous system and manual execution by operators, calculated by averaging over all samples. The resultant autonomy level plans for each autonomy level are enumerated on Table II, along with the measured costs (in seconds) associated with those plans. Additionally, we can construct an estimate of $f_H(j)$ by plotting $\mu(M)$ against $\mu(A)$ and fitting a curve, as plotted on Figure 5.2.

On Table II, we see that the autonomy levels determined by Algorithm 1 have a minimum cost of 42.2 at $A_L = 4$, with the costs increasing from this minima across all other levels. This shows that the algorithm is effecting successive cost improvements around a local minima, with a modest cost gain of 8.5 against the fully autonomous operation, and a substantial gain of 43.7 against the manual implementation.

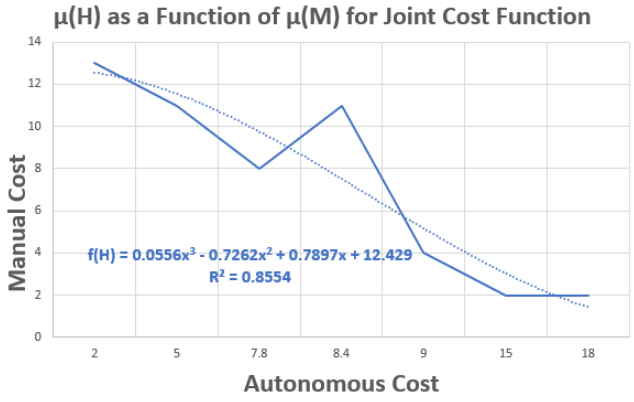

Figure 5.2: Plot of $\mu(M)$ against an index of the corresponding $\mu(A)$ for thew general case, along with the fit curve approximating f_H.

	Autonomous		Manual	
Module	Prob.	Time	Prob.	Time
Target Id	0.90	24s	0.37	36s
Alignment	0.50	17s	0.32	35s
Object track	0.41	13s	0.16	49s
Kinematics	0.67	15s	0.23	47s
Tracking	0.36	24s	0.15	13s
Placement	0.16	11s	0.27	15s
Coverage	0.13	21s	0.15	13s

TABLE 5.1

Average Performance metrics

A_L / Modules	1	2	3	4	5	6	7
Target Id	M	M	A	A	A	A	A
Alignment	M	M	M	M	M	A	A
Object track	M	M	M	M	A	A	A
Kinematics	M	M	M	M	M	M	A
Tracking	M	A	A	A	A	A	A
Placement	M	M	M	A	A	A	A
Coverage	A	A	A	A	A	A	A
Cost Avg	69.5	56.3	47.1	42.2	42.3	44.7	50.7

TABLE 5.2

Derived typical autonomy level plans

These gains and cost profiles are determined for the aggregate data, which prompts the question of the adaptive capacity of the algorithm to individual users. To this end, Table III presents the necessary data for a singular user, in particular one with poor performance compared to the average. This user's manual performance had a total cost of 120.1, fully 40% greater than the average user.

As with the general case, we have the plot of $f_H(j)$, in Figure 5.3, and in Table IV, we have the generated autonomy level plans for this user, which are notably distinct from the case illustrated in Table II. On this table, we also see the same optimization pattern, and significant improvement at the minimum cost over either single-mode operation, with net gains of 71.7, a 60% increase, over the manual-only approach and a marginal advantage

of 2.4 over full autonomy.

Further, under optimization, the low-performer's optima is at $A_L = 3$, with a cost of 48.3, is close to the typical-case optimal cost, with the output autonomy level plan effectively bridging the gap between the low-performing user and the typical user.

Figure 5.4 showcases both cost curves as a function of autonomy level, for which the local minimum is readily identifiable, as in the preceding tables. In particular, the depression of the low-performing user's curve towards the curve of the general case is apparent. We are able to fit high-correlation polynomials to these curves which are locally convex in the fit region, demonstrating that the algorithm generates convex performance curves across autonomy level, as predicted.

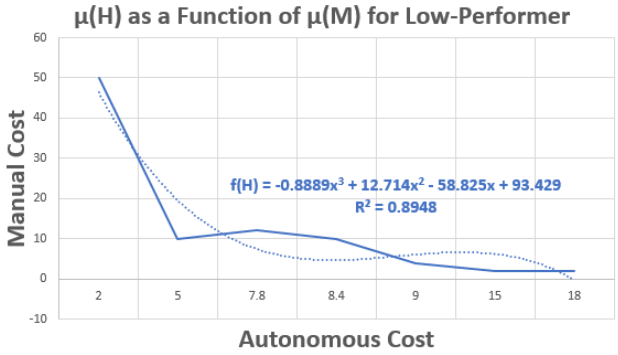

Figure 5.3: Plot of $\mu(M)$ against an index of the corresponding $\mu(M)$ for the low-performing user, along with the approximation of f_H used for analysis.

Module	Prob.	Time
Target Id	0.49	107s
Alignment	0.55	22s
Object track	0.57	17s
Kinematics	0.64	28s
Tracking	0.65	67s
Placement	0.77	17s
Coverage	0.68	33s

TABLE 5.3

Performance metrics of a low-performing user

We can also use the data collected in our experiments to validate the analysis in Sections III. The conditions under which the GAP+ algorithm produces optimal results are

A_L / Modules	1	2	3	4	5	6	7
Target Id	A	A	A	A	A	A	A
Alignment	M	M	M	M	M	M	A
Object track	M	M	M	M	A	A	A
Kinematics	M	M	M	M	M	A	A
Tracking	M	A	A	A	A	A	A
Placement	M	M	M	A	A	A	A
Coverage	M	M	A	A	A	A	A
Cost	78.7	56.9	48.3	48.5	49.0	49.9	50.7

TABLE 5.4

Altered autonomy level plans for low-performing user

monotonicity of μ' under **Theorem 1** via Equation 3.36, and f_H near-monotone within the bounds established in **Theorem 2**. We can calculate the components of each of these relationships directly from our data, establishing that Equation 3.36 holds, and that this problem is indeed optimizable with the GAP+ algorithm.

In Table V, we present the two sides of equation 3.36 for the average and low-performer cases. By examination, we can see that μ' is monotonic, satisfying the first condition. Further, for the average case, we can see that the equation has the minimum difference between the left and right hand sides, 0.12, at $A_L = 4$, identical to the determination of the algorithm shown in Table II. Likewise, for the low-performer, the nearest correspondence is at the determined optima $A_L = 3$, with a difference of 1.83. This shows that for both cases, the measured values in Equation 3.36 are closest to the equivalence condition at the autonomy level determined by the GAP+ algorithm as optimal, and each possesses only one such minimum difference.

There is a discrepancy in this data, of course, in that the values are not precisely equal at the optimum point. However, as we have observed, f_H is not strictly monotonic in this case, suggesting that some measure of deviation will be present. As such, we must rely on the bounds determined in **Theorem 2** to justify optimality, via the inequality in Equation 3.44.

For each subtask, we calculate the change ratio necessary to bring the fit function into

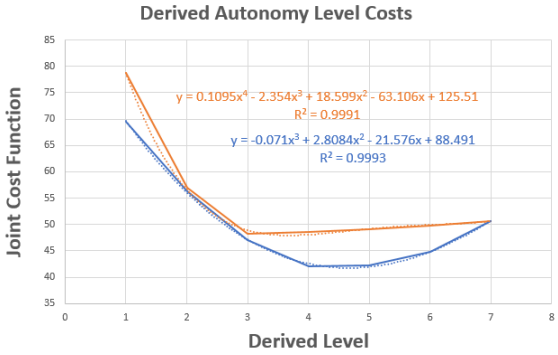

Figure 5.4: Cost curves for both the general case and the low-performance case as a function of the derived level, showcasing the consistent organization of the autonomy levels' performance into convex curves.

compliance with a monotone function. For those modules which are non-conforming, the last row of Table VI contains $\frac{1-r}{r}$, and the corresponding ratio of autonomous to autonomy level plan cost. These values show that in both the general and single-user case the robustness condition is met by at least an order of magnitude. It is worth noting that the average case conforms more closely than the lower performing case, which is to be expected given the higher deviations in the metrics shown in Table V. In particular, we attribute this reduced compliance of the low-performing user's f_H, leading to greater r_j in Equation 11.

	Average Case		Low Performer	
μ'	$\mu ln(f_H) \int \delta\mu$	Δ	$\mu ln(f_H) \int \delta\mu$	Δ
1.33	0.14	1.18	14.93	16.26
1.71	0.26	1.44	11.26	12.97
2.09	0.82	1.26	0.25	1.83
2.47	2.34	0.12	0.56	1.90
2.85	1.31	1.53	0.59	2.25
3.23	1.08	2.14	0.44	2.78
3.61	1.44	2.16	0.62	2.98

TABLE 5.5

Applying Theorem I

In this chapter, we presented the design and analysis of an algorithm for the rigorous selection of levels in a variable autonomy based human intervention autonomy level assignment system. This algorithm was shown to produce optimal assignment of autonomy

	Average Case		Low Performer	
$\mu(M_j)$	$(1-r_j)/r_j$	$\Pi\mu(M/P)$	$(1-r_j)/r_j$	$\Pi\mu(M/P)$
2	0.0014	0.021	0.056	0.035
5	-	0.083	-	0.121
7.8	-	0.187	0.711	0.255
8.4	0.437	0.294	0.696	0.400
9	-	0.408	-	0.555
15	0.151	0.597	-	0.814
18	-	0.825	3.672	1.124
	$9.557 \cdot 10^{-5} < 0.562$		$0.102 < 1.597$	

TABLE 5.6

Applying Theorem II

levels by analysis of the costs used in the assignment algorithm.

To validate our predictions against a real-world model, we tested the algorithm's performance on ARNA, a mobile manipulator, by applying it to optimizing the performance of the complex task of performing a sanitization protocol, decomposed into a hierarchical structure of 7 subtasks. Execution time and transition probability data were taken using a series of trial operations with 15 users, and functions fit to the resulting plots of manual to autonomous cost.

From these data, we constructed the expected cost functions for an average user, and performed optimizations of the autonomy level plans. The resultant variable autonomy levels demonstrated strong optimizing behavior, resulting in a 51% increase in system effectiveness over fully manual operation. To investigate adaptation to specific cases, we examined a single low-performing user and derived a radically different intervention schedule with a 60% cost improvement, bringing their peak performance into the same range as the typical user. In both cases, the costs across of autonomy levels were observed to be convex functions.

We also measured the constituent optimization and robustness conditions for both cases, found the constant conditions we predicted to vary by only a few percent, and the predicted equivalence at the optimal autonomy level being comparably close. Both cases also fell well within the bounds of the robustness condition.

From the observation of strong optimizing performance of the algorithm on both the aggregate and the edge case, and the validation of the predicted cost measurements of the uncertainty and robustness predictions, the data collected in the experimental application are in strong alignment with the theoretical analysis of the algorithm.

However, going forward there are two particular further tasks we would like to pursue. First, in this paper we demonstrated adaptivity by application to a changing user, but application of the VAP+ algorithm in conjunction with probability and cost prediction systems modifying μ in response to other conditions is also possible. This would be an extension that would eliminate the need for situational data collection prior to execution, enabling substantially more flexible operation. In particular, we have been investigating machine learning techniques centered around failure mode prediction to fill this role, and intend later work to study the union of the two systems. The second is investigation of the time-dynamic properties of the VAP+ algorithm. Because it is a greedy algorithm and execution is very fast, it is possible to alter the cost functions associated with the subtasks, perhaps based on sensor information, and calculate autonomy level plans in real time. Operation of the system under these changes is theoretically sound, but need to also be confirmed experimentally.

In this chapter, I outlined the disinfection task for the ARNA robot, presented the task model and experimental procedure for investigating this task, and presented the data and analysis of these experiments. From the data, we can observe the optimizing behavior of the VAP+ algorithm in the context of a problem with multiple paths and a hierarchical structure, as well as see the validation of the conditional boundary and optimality conditions derived in Chapter 3, highlighting that the observed behavior is within the expected operation of the algorithm, and not incidental to its use. We also investigate the effect of the algorithm under changing conditions by examining the utility of the optimization in response to improving a low-performing user's performance relative to the average case user.

CHAPTER 6

ROBOTIC BIN-PICKING WITH VAP+

6.1 Task Description

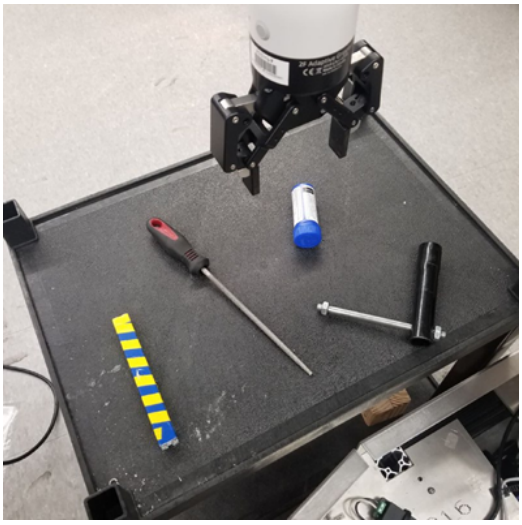

Figure 6.1: Image of the robot workspace for the picking task, including four of the selected items in the 'uncluttered' configuration and the gripper in the initial imaging pose over the work surface

Bin picking is a robotic task domain in which the fundamental problem is, given one or more physical objects, for the robot to approach, grasp, and then move an item. On a fundamental level, the process of picking an item can be broken down into three primary tasks- object localization, trajectory planning, and grasping. Localization is the process by which the target object's position in space is determined, allowing the end effector to move to it. When this position is known, the next step is execution of a trajectory which brings the end effector to within close proximity of the target such that the final step, grasping, can be achieved. Grasping is, in essence, the task of achieving positive, stable contact with the object so that the robot may manipulate it freely.

We chose picking as an example task because it is a well-established and widely studied example of a common task for a robot to perform, but still presents an area where human intervention can be productive for improving efficiency. In particular, it is possible to construct adversarial conditions which degrade robot performance. In our experiments, we focused on two difficulty regimes: uncluttered picking and cluttered picking.

Uncluttered picking presumes that all potential targets in the robot's field of view are physically separated, whereas cluttered pricking presumes the presence of contact and potentially occlusions. This pair of domains allows us to investigate the impact of changing problem complexity while still maintaining the fundamental structure of the task.

For the purposes of the variable autonomy task planner, we decompose the task into five total subtasks. The task begins in the 'start' state, with the robot arm posed over the workspace as seen in Figure 6.1, and terminates in the 'goal' state, which is positive contact with the object, verified after completion of the grasping subtask by attempting to lift the object.

Each of the subtasks may be executed autonomously, or manually by an operator through a user interface, as decided by the VAP algorithm. These tasks are:

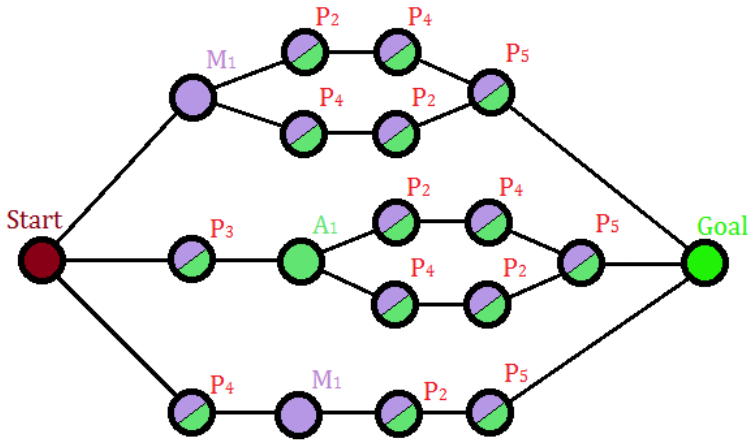

Figure 6.2: Illustration of the task graph for the picking task used in these experiments. In this representation, the green nodes are autonomous modules and the purple are manual. Modules which can, at any specific stage, be executed by either module are represented by split-color nodes.

P_1: Positioning of the End-effector in the XY Plane, aligning the end effector with

the target object

P_2: Rotation of the end-effector in the Θ_Z plane, to align the gripper with the target

P_3: Selection of object in the field of view for tracking, either directly by a human, or using the autonomous tracking algorithm to maintain object identification during movement

P_4: Setting height of the end effector to the appropriate level at which a grasp can be executed

P_5: Executing the grasp of the object and successfully lifting the object for placement

These segments need not be expressly executed in a fixed order, as we will see in Section III(B), and the structure of the state machine used to implement the variable autonomy algorithm is constructed to accommodate the wide variety of strategies implemented by different users to accomplish the task. However, there are hierarchical relationships embedded in this subtask decomposition.

Firstly, P_5 is necessarily the terminal subtask as it is both necessary for task completion, and effects the transition to the goal state, with is positive contact with the object. Additionally, the autonomous implementation of P_1 requires a target be selected, and thus A_1 must be preceded by P_3, either manual or autonomous. Finally, P_2 and P_4 are generally inter changeable, but both must precede P_5.

The VAP algorithm is implemented on a state machine which controls the transitions between each subtask, with the autonomy level plan output of the algorithm specifying which subtasks are performed autonomously and which are performed manually. In constructing the state machine, we must specifically account for all the varying approaches that may be implemented to achieve the task.

Figure 6.2 presents a diagram which represents the task graph T_G on which state machine is implemented. It is worth noting that this T_G is reduced to represent the observed strategies only- it is possible to begin the task with P_2, however no user was observed to select this, and autonomous implementation using this approach were unilaterally ineffective, thus for clarity we do not implement it.

It is worth noting that the subtasks are placed throughout the graph, applicable at

different stages, depending on the exact strategy in play. For instance, one user may be more inclined towards posing the end-effector height before rotating to match the object, versus another user may prefer to rotate and then lower the gripper. Both possibilities are accounted for so as to allow the planner to implement the overall best policy.

6.2 Experimental procedure

In this section, we discuss the experimental structure used to evaluate the performance of the algorithm, especially with regards to the commonalities and differences with respect to the implementation in the prior system. In particular, the new robot and software are systematic improvements over the prior application case, for which the improvements substantially increase the capability to probe performance aspects of the VAP algorithm. However, we have replicated the task being performed as closely as possible, so as to make comparisons between the human judgement method of selecting autonomy level and the autonomous selection of autonomy level as resilient to experimental variations as possible.

	Uncluttered Autonomous		Uncluttered Manual		Cluttered Autonomous		Cluttered Manual	
Module	Prob.	Time	Prob.	Time	Prob.	Time	Prob.	Time
P_1	0.7	2.8s	0.5	3.2s	0.7	5.9s	0.7	4.2s
P_2	0.9	4.9s	0.8	4.3s	0.6	3.1s	0.8	5.8s
P_3	0.7	5.4s	0.7	3.2s	0.5	3.0s	0.6	5.9s
P_4	0.9	4.1s	0.8	5.6s	0.6	2.5s	0.7	6.2s
P_5	0.8	1.9s	0.8	5.2s	0.7	2.9s	0.8	4.1s

TABLE 6.1

Average Performance metrics for Cluttered and Uncluttered tasks, Second trial

6.2.1 Experiment Design

Throughout our experiments, we have each of the 25 subject perform three trial runs from task initiation to completion. As mentioned before, all previous experiments included at least one first 'familiarization' trial, which acted as the user's introduction to the UI and the task itself. We selected three trials on basis of both the observations made with

	Uncluttered Autonomous		Uncluttered Manual		Cluttered Autonomous		Cluttered Manual	
Module	Prob.	Time	Prob.	Time	Prob.	Time	Prob.	Time
P_1	0.7	2.8s	0.8	2.4s	0.7	5.9s	0.8	4.3s
P_2	0.9	4.9s	0.9	3.0s	0.6	3.1s	0.8	2.6s
P_3	0.7	5.4s	0.7	1.9s	0.5	3.0s	0.8	5.0s
P_4	0.9	4.1s	0.9	6.3s	0.6	2.5s	0.9	4.7s
P_5	0.8	1.9s	0.8	4.2s	0.7	2.9s	0.7	4.5s

TABLE 6.2

Average Performance metrics for Cluttered and Uncluttered tasks, Third trial

repeat users in experiments involving other tasks, and pilot experiments for this specific task. What these pilot studies revealed is that user's first runs are, in general, very low performing, as would be expected. However, their later runs are typically substantially better, with the third trial being consistent enough with their longer-term performance that autonomy level selection by the VAP algorithm rarely changes after the third trial.

In these trials, the robot is set to perform individual subtasks, with the requisite cost data being comprised of the total execution time, measured by execution software packaged along with the primary modules in the system state machine; and failure rates measured by number of hard faults- physical situations in which the entire process must be restarted, which applies an error count to all modules used up to the time of the fault. Hard faults include failure to successfully grasp the object, collisions of the robot with the workspace or non-target items, and tracking failure in which the vision system can no longer perceive the tracked object. We also include soft errors- times in which the user must re-attempt a subtask. Note that the autonomous modules are subject only to hard faults, as the robot will not autonomously re-attempt a subtask.

We have three sets of experiments in total, those to determine the performance of the autonomous systems, those to measure human performance across a range of subject, and finally the latter phase of testing the predicted autonomy level plans against their actual execution. The autonomous performance trials are entirely for the purpose of acquiring data for the VAP algorithm. The human trials perform the same function, except with the

objective being to determine the cost data for the manual modules. Finally, the tests against predicted performance implement one of the variable autonomy level plans, to measure execution time for comparison to the predicted times for the VAP algorithm's output.

	Trial 1		Trial 2		Trial 3	
Autonomy Level	Pred.	Meas.	Pred.	Meas.	Pred.	Meas.
1	15.3	15.0	15.8	16.4	15.3	16.2
2	13.6	13.5	14.6	14.7	13.3	15.5
3	12.4	12.5	14.2	13.9	11.5	13.2
4	12.1	12.6	14.9	12.8	11.6	11.7
5	13.1	13.7	15.7	13.0	13.2	16.2
6	15.6	15.8	16.6	15.0	15.6	16.5
$Av.Error:$		0.6%		-7.6%		9.5%

TABLE 6.3

Predicted and Measured Autonomy level costs for uncluttered picking task

Each trial begins with the robot end effector posed as illustrated in Figure 6.1. In this pose, the camera frame includes the entire workspace. An object is selected, either by human perception directly or using the internal tracking mechanism for the autonomous module, which picks the nearest item to the center of the visual field. The selected object is then approached, using one of many strategies possible, as seen in Figure 6.2. Once the approach and alignment to the object is complete, grasping is effected by closing the gripper and lifting the item. If the robot can, at this stage, successfully lift the object, a success is registered, otherwise a failure, and the trial is complete.

One qualitative note is that most users typically express only one 'strategy', or more precisely, attempt to execute one specific path through the task graph. For instance, a common approach for human users was to position the camera over the target object at the initial height, rotate the gripper to match the object orientation, and then drop and grasp the object. An alternative, also fairly common, was to position the camera, lower the gripper, and then rotate. This latter strategy tends to result in overall improved performance, as position errors associated with rotation are minimized when closer to the object.

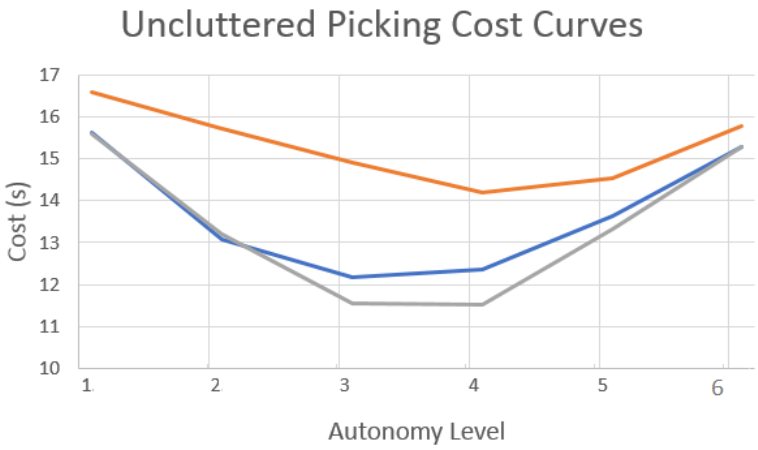

Figure 6.3: Cost curves derived from the VAP algorithm, for the uncluttered picking case and each of the three trials, plotted against autonomy level

Throughout all three trials, the majority of users implement only one such strategy to execute the task. One anomalous user, however, executed the task initially using the 'drop, then rotate' approach. However, upon her second trial, elected instead to lower the gripper near the object and use it to rotate the object such that she could move the gripper around the object and grasp it. This procedure, while not the most optimal process, presents a challenge to the algorithm, as it is dramatically different from the typical process. In this case though, the algorithm still adapted and produced a more efficient path, as the task graph, 6.2 accommodates the altered process order.

6.3 Experimental Results

Our first task is to collect the necessary performance data for the VAP algorithm. This data is collected across the three trials independently, so that we can calculate the autonomy level plans across each trial and compare the performance of the algorithm as the human adapts to the problem. Further, as the original variable autonomy experiments were performed in both cluttered and uncluttered contexts, we also perform our analysis over each of these problem domains.

Table I presents the requisite data, including the probabilities of successful completion of each module and the average time across all users who performed these subtasks,

at trial 2, after the acclimatization trial. Table II presents the same data as collected at the third trial. Note that the autonomous measures for uncluttered and cluttered picking are the same, across these tables as the autonomous system is not changing average performance of individual subtask modules, only adapting the choices in autonomy planning made in response to the user's performance changes.

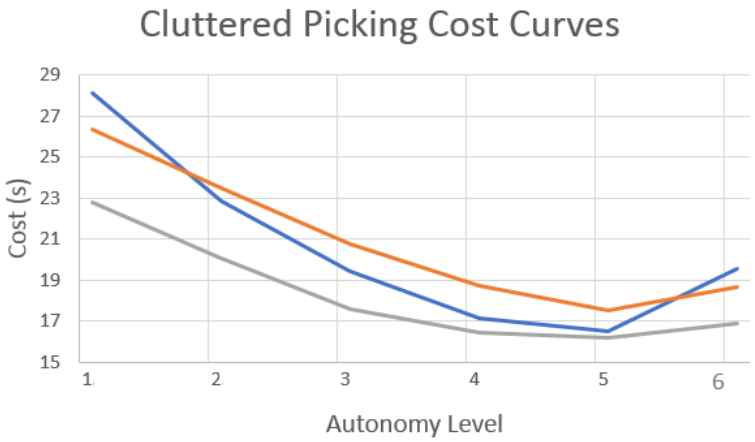

Figure 6.4: Cost curves derived from the VAP algorithm, for the cluttered picking case and each of the three trials, plotted against autonomy level

	Trial 1		Trial 2		Trial 3	
Autonomy Level	Pred.	Meas.	Pred.	Meas.	Pred.	Meas.
1	19.6	19.0	18.7	21.2	16.8	16.6
2	16.5	15.5	17.5	15.3	16.2	15.7
3	17.1	14.5	18.7	19.1	16.5	14.3
4	19.4	20.1	20.7	20.4	17.6	15.1
5	22.8	21.1	23.5	20.7	20.1	17.6
6	28.1	30.5	26.3	27.7	22.8	22.7
Av.Error :		-4.1%		-1.8%		-8.5%

TABLE 6.4

Predicted and Measured Autonomy level costs for cluttered picking task

Further, when we examine the manual performance metrics, we can see that while the execution times remain comparable, generally within ±15%, the probabilities of success are consistently improved, supporting the observation that the learning phenomena is indeed occurring between the second and third trials.

Once we have this data, we can calculate the costs over the planning graph as discussed in Section II(B), and apply the VAP algorithm to the job of determining autonomy level plans. For each of the 6 autonomy levels, we produce a predicted cost for that corresponding plan, and Figures 6.3 and 6.4 showcase the cost curves for the derived plans, plotting the cost in seconds as a function of autonomy level.

Examining these graphs, we can make three key observations. The first is that in both cases, cluttered and uncluttered, all the plan curves are convex, as a direct consequence of the VAP algorithm's construction. This allows us to readily optimize on this curve by selecting the single minimum cost autonomy level.

Secondly, we can see the pattern of optimization working across trials. As the user becomes more efficient at performing the subtasks, the overall height of the curves decreases, corresponding to consistent improvement of the performance at all autonomy levels, and in particular consistent decrease in the cost of the optimal points. This shows that as the user learns, the VAP algorithm is both able to adapt to, and make use of, the improved human performance.

Finally, we can identify, for each trial, the optimal autonomy level itself. This is the central comparison to the original experiment, in which we found that the consistent human-selected autonomy levels, which were pareto optimal, were $A_L = 3$ for the uncluttered case, and $A_L = 5$ for the cluttered case.

For these experiments, our results hew close to the prior experiments, with a few exceptions. For the cluttered picking case, the optimum plan was found to be $A_L = 5$ across all trials, fully in keeping with the prior results. For trials 2 and 3 in the uncluttered case, we found consistent results as well, with $A_L = 3$ being the optimum points. However, for the initial trial (which we did not explicitly measure in the original experiments), $A_L = 4$ was the lowest-cost autonomy level. While this shows, as with the shifting of the cost curves, that the VAP algorithm is indeed adapting to the human's development, but it is an anomalous result with respect to our prior experiment.

It would be natural to presume that this was the result of human learning applied

in the original experiment without being measured, however it is curious that we do not see similar phenomena with the cluttered case, an expectation we would have given that the cluttered case is higher complexity, as indicated by the lower success probabilities across the full battery of cluttered tests compared to the uncluttered tests.

We can, however, observe some relevant comparisons that shed light on this discrepancy. First, the curves for the uncluttered state are substantially less steep than those of the cluttered case, and the minima are substantially closer in cost to their neighbours. In Trial 1 of the uncluttered case, for instance, $A_L = 3$ has a cost of 12.4s, and $A_L = 4$ has a cost of 12.1s. What this may represent is that the uncluttered case is more susceptible to changes in autonomy level because the span of costs is lower on the whole, making the impact of the learning more observable from the VAP algorithm's perspective, such that the change due to human variance is in fact more substantive in the lower complexity problem.

In addition to measurements in the predicted cost curves as determined by the algorithm, it is also critical to the validation to compare the actual performance when operating under the autonomy level plans for each level. To compare each level, we performed a battery of experiments in which each autonomy level plan was executed with a human operator, and measure the actual completion times for these trials. We compare each trial by having users implement different plans at each level, such that the trial 2 measured value is that user's second trial.

The data from these experiments are plotted on Tables III and IV, across each autonomy level and all three trials for the cluttered and uncluttered domains. In all cases, the average deviation of the measured cost from the predicted cost is within 10% of the measured cost, taken to be the ground-truth value. As such, we can see that the predictions are, in general in alignment with the predicted costs. Further, and more importantly, the general pattern of minimization around a single autonomy level is present, albeit with some slight variation. This shows that the trend towards optimization is still present, even under perturbations.

6.4 Analysis

In this chapter, we presented an evaluation of our VAP algorithm, designed to identify optimal levels of autonomy by selecting whether subtasks in a specific task graph are accomplished by a human or a robot agent. In particular, we sought to recreate the conditions of the original experiment from which the inspiration to develop this algorithm was drawn and determine whether or not the patterns evident in the human judgement being leveraged for autonomy level selection in the original system were adequately replicated in the VAP algorithm

Towards this end, we developed a comparable bin picking task based on the original task and implementing similar systems for visual processing, grasping, and trajectory generation. Further, we selected our experimental conditions to mirror those of the original experiment as closely as possible, using visual feedback through a user interface and a set of 2-dimensional control regimes, along with the same style of joystick for control. We also designed the state machine on which the robot operates to match the subtask allocations made on in the prior work, so that the same process relationships were being tested.

We performed three batteries of experiments to produce results for comparison to the original case, as well as determine additional information about the operation and performance of the algorithm, based on observations made during the original experiments. Each set of experiments had three trials each, to allow for study of adaptation to human learning, and covered the same two complexity classes investigated previously- cluttered and uncluttered picking.

The first set of experiments determined the necessary data pertaining to the autonomous portions of the robot system to execute the VAP algorithm, and the second to do the same for human users. The latter set of experiments used the autonomy level plans derived from this data to produce a comparison set of costs using the plans in practice, so that the predicted costs could be validated by experiment.

In analysing out data, we were able to make several conclusions about the VAP algorithm. First, we were indeed able to confirm that the VAP planner derived the same

autonomy levels in most cases as the human-choice based sliding scale autonomy. We observed one exception to this, and by observation of the data related it to the sensitivity of the system, on basis of the small magnitude of cost difference between the non-matching autonomy level in the anomalous case, as well as the absence of this difference in the more complex cluttered case.

We also examined the effect of human learning on the planning algorithm, plotting the cost curves for the derived plans at each trial. These plots showed that, as the human learned the task and UI, the VAP algorithm adjusted to both keep the plans optimal and generally improve the cost by leveraging improvements in human performance, with the curves decreasing in overall average cost, and the costs of the optimal solutions also decreasing.

Finally, we use the battery of tests at the autonomy level plans determined by the algorithm to compare the actual performance of these plans, when executed, against the predicted cost. With this comparison, we see that the measured costs and actual costs for these plans are very close, and that the general trend of convexity of the cost curves, and thus optimizability, also remains present.

On the whole, these observations and inferences do indeed support the conclusion that the VAP algorithm successfully delivers on its design, generating autonomy level plans which, without human selection, emulate the choice mechanisms at play when humans use their judgement to select autonomy level. By testing in an experimental context made to emulate the prior work, we have derived the same results pertaining to the optimal autonomy level selection as human operators.

One specific limitation of the VAP algorithm is that it presumes two kinds of modules- manual and autonomous. In practice, this really means modules which are either autonomous, or dominated by manual control, but the manual modules still implement human interface controls over top of a basic control system. To wit, the human controls cartesian positioning of the robot, but the embedded controller still performs the inverse kinematics and joint level velocity control. This leads directly to the hypothesis that one

might have multiple degrees of sub executive level of control within the 'manual' modules, potentially presenting a question of level of autonomy within them. One of our goals, going forward, is to incorporate this potentiality into the algorithm, such that it is possible to solve the problem of variable autonomy at the subtask level, without having to further decompose the subtasks into multiple parallel tracks representing each level of autonomy at the low level, or perhaps even the continuous optimization problem of incorporating fully variable autonomy at each subtask.

In this chapter, I applied the VAP+ algorithm to a bin picking task designed to emulate the initial experiments with the Baxter robot on thr ARNA system. Through these experiments, we confirmed that the autonomy level assignment and selection as performed by the VAP+ algorithm is emulating action of the human operators in the early trials, determining the same levels of autonomy in the new application as in the old. Further, we use these experiments to confirm that the predicted cost curves produced by the VAP+ algorithm align with the measured cost curves, as well as investigate the VAP+ algorithm under conditions in which temporal variance is present due to human learning.

CHAPTER 7

CONCLUSIONS & FUTURE WORK

To conclude, we will be discussing the results thus determined, the implications for future work, plans for these follow-up efforts, and finally a summary of the material contributions thus far produced.

7.1 Summary of Results

Results thus far are of four types: The algorithms as derived; the associated proofs; data supporting the efficacy of the VAP algorithm; and the developed support systems for implementing the VAP experiments.

7.1.1 Algorithms

In this work, we have presented two algorithms- the VAP algorithm which is designed to formalize a rigorous mathematical definition for autonomy level based on measurable parameters of system operation, and the VAP algorithm which extends the action of this metric-based systematic system into more complex task spaces.

The VAP algorithm was derived by examining a universal property of planning systems- success probability. Under this basis, we were able to produce a ranked-order system in which the derived autonomy level parameter was both determined by the ordering and the system performance could be expressed in terms of a readily observable metric. This allowed for the autonomy level parameter to be linked to the uncertainty in performance, and thus by virtue of being a human-controlled component of the system, allowed for use of human perception to achieve effective performance under changing conditions.

By examining data from the VAP experiments and identifying a candidate opti-

mization heuristic from the apparent effects of the human choices on optimum selection, we were able to design the VAP algorithm, which implemented an expected cost formulation to analyze changes across a planning graph, and subsequently select a sequence of module replacements which achieve a locally optimal plan. The advantage of the VAP algorithm being compatibility with planning outside of the restricted sequential assumptions underlying the VAP algorithm.

7.1.2 Proof

To establish efficacy of the VAP algorithm, we performed an analysis based on the selection of a metric. Using this selection, we identified a relationship model between the autonomous and manual modes of operation so as to parametrize the relationship in terms of the machine module performance when sorted.

This relationship allowed us to establish two critical facts. First, the conditions under which the selected autonomy level definition can be known to be correlated with an optimizable performance function, and as a corollary, the development of a rigorous backing for the nature of the cross-sectional optimum being tied to the inverse relationship between human and machine competencies.

Second, we demonstrated that there is a positive conditional range associated with disturbances from the strict inverse functional relationship between manual and autonomous performance under which a local optimum in the autonomy level exists. This analysis was performed by hypothetical modifications of the performance function to an envelope function bounding the curve while satisfying the monotonicity requirements of the earlier analysis.

By virtue of these two properties, we established that out chosen model for autonomy level was both quantitatively linked to optimal performance under conditions, the conditions under which this relationship holds, and that as a result the effectiveness as a sliding scale system is predicated only on the performance of the effectively outsourced task of operator prediction skill.

For the VAP algorithm, we modified our construction of the selection criteria so as to work with an expected cost from the starting state to the goal state, and from that designed by construction an algorithm which selected a local minimum expected cost within a neighborhood of the local modules under the operation of replacement. This construction guarantees that the resultant selected level of autonomy will be an optima, and further alleviates the need of human selection of the autonomy level, though a set of fully-ordered levels are also defined in service to this goal.

7.1.3 VAP Experiments

In this section, we discussed the impact of human and robot collaborative interaction on the efficiency and efficacy of task completion, with a specific emphasis on the relation between the relative advantages between the human and machine. Furthermore, we postulated that the effectiveness of a shared control system may be context-dependent, and as such, a system which adapts the level of autonomy would out perform a system with a fixed level of autonomy. We concurrently proposed that this adaptation task be assigned to the human as a sliding scale input, resulting in the Variable Autonomy Planner for augmenting an autonomous system. By contrast to traditional variable autonomy systems, we developed a quantitative metric for assignment of subtask execution modules to autonomy levels, in which the level of autonomy corresponded to the inclusion of autonomous modules by rank, according to empirically determined failure modes.

To validate the predictions of the analysis underwriting the VAP algorithm, we performed experiments on a common picking task under changing complexity conditions so as to prompt alteration of the autonomy level by the operator. By this means, we were able to investigate the effectiveness of the autonomy level assignment method embedded within the VAP algorithm.

To investigate the performance of our algorithm, we posed an example pick-and-place task and devised a set of scenarios designed to imitate such variable conditions. We tested this system in the typical fixed mode, as well as the adjustable mode, and from

these experiments observed that under this assignment model, user selections of A_L did indeed correspond to the optimal performance balance for the tasks, which indicates that the failure probability metric is an efficacious proxy for task complexity.

In these experiments, we observed firstly that there was a rough inverse relationship between manual and autonomous module performance curves, which established the necessary baseline for our theoretical predictions to be considered testable in this case. Further, by testing all combinations of autonomy level and complexity class, we were also able to identify the optimal autonomy level in each problem class without resorting to ad hoc evaluation by a human designer.

When allowing the operator to have control of the sliding scale parameter, then, what we observed was that they consistently selected the parameter level which yielded the lowest in-class error rate at the highest net speed. This, combined with the knowledge that picking tasks are simple enough in execution to be accurately predicted by an operator, provided strong evidence for the efficacy of the VAP algorithm as a means of implementing sliding scale autonomy with a repeatable mechanism for level assignment without recourse to human design choices.

Additional future experiments investigating non-environmental aspects operation, such as human factors, would be an important step towards generalizing our results. On particular case is the investigation of the apparent small advantage in speed for the VAP execution batteries, which seems anomalous to the preformance curve defined by Fitts' Law. We also plan to investigate context-dependent changes in non-complexity oriented changes in task domain, such as single-module failure rate increasing factors. One example of this would be changes in light conditions, which almost exclusively impacts the object recognition module. A solution to this variability may be inclusion of multiple machine modules for certain subtasks, integrated to the probabilistically ranked A_L by automatic scene analysis.

7.1.4 VAP+ Experiments

In preparation for application of the derived VAP algorithm to a real problem for validation, we have implemented several subsystems for the achievement of a complex non-sequential task suitable for VAP processing and incompatible with the VAP algorithm via the ARNA sanitizaiton robot.

This task seeks to perform sanitation of a target, requiring submodules capable of recognition, tracking, navigation, kinematic control and manipulation of a tool, all combined with potential for multiple interactions between tasks depending on outcomes. The resultant planning graph is thus not capable of being resolved to a chain.

For each of the seven described modules, we have designed an autonomous and a manual system, with each being independent up to the transitions. Further, a primary controller which performs the VAP planning steps via a primary state-machine controller, as described for the VAP algorithm, has also been prepared for the robot.

In this battery of experiments, we presented the design and analysis of an algorithm for the rigorous selection of levels in a variable autonomy based human intervention autonomy level assignment system. This algorithm was shown to produce optimal assignment of autonomy levels by analysis of the costs used in the assignment algorithm.

To validate our predictions against a real-world model, we tested the algorithm's performance on ARNA, a mobile manipulator, by applying it to optimizing the performance of the complex task of performing a sanitization protocol, decomposed into a hierarchical structure of 7 subtasks. Execution time and transition probability data were taken using a series of trial operations with 15 users, and functions fit to the resulting plots of manual to autonomous cost.

From these data, we constructed the expected cost functions for an average user, and performed optimizations of the autonomy level plans. The resultant variable autonomy levels demonstrated strong optimizing behavior, resulting in a 51% increase in system effectiveness over fully manual operation. To investigate adaptation to specific cases, we examined a single low-performing user and derived a radically different intervention schedule

with a 60% cost improvement, bringing their peak performance into the same range as the typical user. In both cases, the costs across of autonomy levels were observed to be convex functions.

We also measured the constituent optimization and robustness conditions for both cases, found the constant conditions we predicted to vary by only a few percent, and the predicted equivalence at the optimal autonomy level being comparably close. Both cases also fell well within the bounds of the robustness condition.

From the observation of strong optimizing performance of the algorithm on both the aggregate and the edge case, and the validation of the predicted cost measurements of the uncertainty and robustness predictions, the data collected in the experimental application are in strong alignment with the theoretical analysis of the algorithm. This set of confirmations validates the expected operation of the VAP+ algorithm, allowing us to move on to confirmation of the design principles based on the initial picking experiments with VAP.

Towards this end, we developed a comparable bin picking task based on the original task and implementing similar systems for visual processing, grasping, and trajectory generation. Further, we selected our experimental conditions to mirror those of the original experiment as closely as possible, using visual feedback through a user interface and a set of 2-dimensional control regimes, along with the same style of joystick for control. We also designed the state machine on which the robot operates to match the subtask allocations made on in the prior work, so that the same process relationships were being tested.

We performed three batteries of experiments to produce results for comparison to the original case, as well as determine additional information about the operation and performance of the algorithm, based on observations made during the original experiments. Each set of experiments had three trials each, to allow for study of adaptation to human learning, and covered the same two complexity classes investigated previously- cluttered and uncluttered picking.

The first set of experiments determined the necessary data pertaining to the autonomous portions of the robot system to execute the VAP algorithm, and the second to

do the same for human users. The latter set of experiments used the autonomy level plans derived from this data to produce a comparison set of costs using the plans in practice, so that the predicted costs could be validated by experiment.

In analysing out data, we were able to make several conclusions about the VAP algorithm. First, we were indeed able to confirm that the VAP planner derived the same autonomy levels in most cases as the human-choice based sliding scale autonomy. We observed one exception to this, and by observation of the data related it to the sensitivity of the system, on basis of the small magnitude of cost difference between the non-matching autonomy level in the anomalous case, as well as the absence of this difference in the more complex cluttered case.

We also examined the effect of human learning on the planning algorithm, plotting the cost curves for the derived plans at each trial. These plots showed that, as the human learned the task and UI, the VAP algorithm adjusted to both keep the plans optimal and generally improve the cost by leveraging improvements in human performance, with the curves decreasing in overall average cost, and the costs of the optimal solutions also decreasing.

Finally, we use the battery of tests at the autonomy level plans determined by the algorithm to compare the actual performance of these plans, when executed, against the predicted cost. With this comparison, we see that the measured costs and actual costs for these plans are very close, and that the general trend of convexity of the cost curves, and thus optimizability, also remains present.

On the whole, these observations and inferences do indeed support the conclusion that the VAP algorithm successfully delivers on its design, generating autonomy level plans which, without human selection, emulate the choice mechanisms at play when humans use their judgement to select autonomy level. By testing in an experimental context made to emulate the prior work, we have derived the same results pertaining to the optimal autonomy level selection as human operators.

One specific limitation of the VAP algorithm is that it presumes two kinds of

modules- manual and autonomous. In practice, this really means modules which are either autonomous, or dominated by manual control, but the manual modules still implement human interface controls over top of a basic control system. To wit, the human controls cartesian positioning of the robot, but the embedded controller still performs the inverse kinematics and joint level velocity control. This leads directly to the hypothesis that one might have multiple degrees of sub executive level of control within the 'manual' modules, potentially presenting a question of level of autonomy within them. One of our goals, going forward, is to incorporate this potentiality into the algorithm, such that it is possible to solve the problem of variable autonomy at the subtask level, without having to further decompose the subtasks into multiple parallel tracks representing each level of autonomy at the low level, or perhaps even the continuous optimization problem of incorporating fully variable autonomy at each subtask.

7.2 Limitations

As the algorithms currently stand, there are three main limitations of the VAP+ algorithm which restrict its application and bear further investigation for improvement:

(1) Optimality Conditions

Currently the most substantial limitation are the optimality conditions expressed via Equation 3.36. While the later derivation of Equation 3.44 does provide a relative range of allowable deviations from the strict monotonicity conditions, it is still a fairly strict requirement. One approach that may help to eliminate the need for this conditional in the argument is a modification of Equation 3.29, the proxy function used to examine the cost-change landscape for the joint cost model problem. This expression is, in essence, similar to a Lyapunov function- it is a representative form that bounds the conditions associated with the behavior of $\mu(S)$. However, it is also fairly vague, and one might suspect that a more nuanced model may yeild more relaxed conditions, however as yet we have not identified such a suitable function which is analytically tractable to prove either convexivity of $\mu(S)$, or to replace 3.29 in the current extrema-based proof structure.

As a further consideration, the monotone conditions we elect to use do not expressly contain all possible optimizable models- it is possible to consider other structures for the form of 3.36 besides monotonicity which produce only one extrema, but out condition does not address this. Thus our optimization proof presents a sufficient, but not necessary condition, at least hypothetically. It may be possible, as mentioned above, to construct an alternative form of bound equation which also allows for an optimality policy which captures a greater proportion of all possible optimal systems. It is, however, more likely that construction of a proof of necessity would be more likely rooted in the combinatorial structure of the task graph analysis. We suspect that the most likely outcome of work towards this condition is an algorithm which analyzes a specific task graph to calculate a measure of optimizability, rather than a validity check on the data.

(2) Task Graph Structure

Throughout this work, we have presumed a directed, acyclic graph as the task accomplishment model. This obviously presents a specific limit, as processes with potential cyclical executions are excluded. It is possible to, on some level, adapt the algorithm to a level of repetition by expanding the task graph to include multiple nodes, though this does increase the size of the graph itself substantially, with each ordered node inclusion multiplying the number of paths in the subsequent side of its successor paths by its out-order.

The primary issue in involving cycles within the graph is the question of expected cost. Formally, cycles would invalidate Condition 3, however a more intuitive note is that repeating a process on a cycle indefinitely leads to infinite cost. In part, we have considered the impact of some repetition in the form of the failure probability, which measures the effect of failing a subtask, which at later stages may amount to repeating a large number of prior subtasks and thus be similar to a cycle, but ultimately presumes a return to the start state, or an implicit increase in expected cost for individual subtasks.

It is somewhat difficult to conceptualize a means for eliminating this, however one might conjecture that calculation of step wise expected costs over only the subtask costs on a minimum spanning tree from each node to the goal would be a potential resolution.

If that were the application state, then the model for DAGs would be a special case where the MST was identical to the graph itself and resolve to the current VAP+ algorithm. In non DAG cases, then, it would calculate a limited cost under the presumption that loop-related expected costs are embedded in the measurements taken throughout the experiments. However, it would be a matter of experimental evaluation to determine if this adequately represents the actual way that cycles are executed in real world systems.

(3) Environmental variance

In this construction, we have presumed that there is variance among human performance which is the primary motivator of deviations in plan costs over users. It is also possible to consider the cases in which there is variance within the autonomous system which is strictly environmentally based. For instance, vision algorithm performance may shift as a result of lighting conditions. This change amounts to a shift in the positioning of $\mu(A_i)$. While these shifts, if they preserve the necessary relationships for f_H, will still be optimizable, it does bring up the point of thr sampling being skewed between conditions influencing overall performance.

For instance, say the measurements taken for $\mu(A_i)$ are dependent on the environment and there are two conditional cases which are functionally the cause of the variance, situations A and B. If we measure the autonomous performance under only situation A, then the VAP+ algorithm is likely to produce poor results during operation in situation B, and vice versa. However, if we take a sampling over both situations for $\mu(A_i)$, we can conceive of a situation where the resulting average is partway between two very different costs, and thus heavily inaccurate for assigning the autonomy levels in either situation.

The natural response to this, given the speed of computation for the VAP+ algorithm, is to connect the costs being used at any instance of VAP+ level assignment to observed conditions. The ways in which this connection could be made are numerous and beyond the scope of this work, but utilization of some form of level calculation would allow for dynamic adaptation beyond that seen to variant human users and complexity classes and extend the utility of the VAP+ algorithm. One might even hypothesize an adaptive

system which is based not on accurate prediction of $\mu(a_i)$, precisely, but rather based on optimizing performance of the VAP+ algorithm by minimizing variance in $\mu(A_i)$, such that the presumption of dominance of human uncertainty is always a true and strong proposition.

7.3 Future Work

The future tasks for this work are oriented around performing the experiments to validate the VAP algorithm by emulating the procedure used for creation of the comparisons which, for the VAP algorithm, demonstrated the effectiveness of the underlying optimization principles.

In these experiments, the same panel-based process will be used to examine each potential level of autonomy as defined by the algorithm derivation, in which is included the level self-identified as the local optimum in expected cost. This local point can be compared statistically using the measured actual costs of executions in each mode to the same experiment performed at other, predicted non-optimum levels.

In addition, the VAP algorithm root data includes measures for all potential transitions, and thus we can look further into the relative effectiveness of the heuristic model which produced the expected-cost minimizing heuristic by a fully combinatoric examination of all viable paths through the planning graph to select the theoretical global minimum.

By further applying the measured variances to the transition probabilities and costs, this exploration can be expanded to account for the variances across the measurements, which will allow a determination of the ranges across which environmental effects to the overall planning may be significant. The sensitive modules identified at this stage may then be artificially altered to induce a planning change. Experiments under this performance alteration will enable the investigation of the fitness of the adaptive properties embedded within the expected cost metric.

Knowing the degree to which the conjoining of a predicting system to the conditional probabilities used in the cost calculation will establish the viability of the system as a whole as an adaptive variable autonomy planning algorithm utilizing rigorous metrics. With this,

we will have an intervention planner which possesses establish performance guarantees with validation, and which not only quantifies autonomy levels without manual definition, but also optimizes the autonomy level without leaning on human perception, tidily resolving the outstanding issue described in Chapter I.

7.3.1 Cost function adaptations

The simplest way to effect the process of selecting the autonomy level is to couple costs to some kind of input data. This may be sensor data that determines an objective function from measurable properties (say, a system that calculates expense of raw materials by weight) or even a point of access for human input, in which a user makes an estimate based on observations for the planner to make use of.

The root factor in cost is that it will apply within the planning algorithm without regards to structure, so long as the mapping from input to cost generates the same type of cost as used throughout the system, consistency will be preserved. This means that one might design a cost function with weights set by sensor data, or even that an operator might toggle between cost estimator methods in a terminal. Functionally, the central point is that the inherent agnosticism of the joint expected cost metric will accept any scalar data calculation.

7.3.1.1 Probability adaptations

While cost-based changes are useful, particularly considering integration with HRI systems for identifying human factors costs, the most potent means of adjusting system operation is in the probability functions.

Figure 7.1 illustrates the fundamental character of this kind of adaptive system, in which some form of data input has a mapping function onto the conditional probabilities. In many ways, this is similar to the cost function alterations, however the notable case is that cost functions are often ad hoc or fixed, but probabilities change frequently based on small shifts.

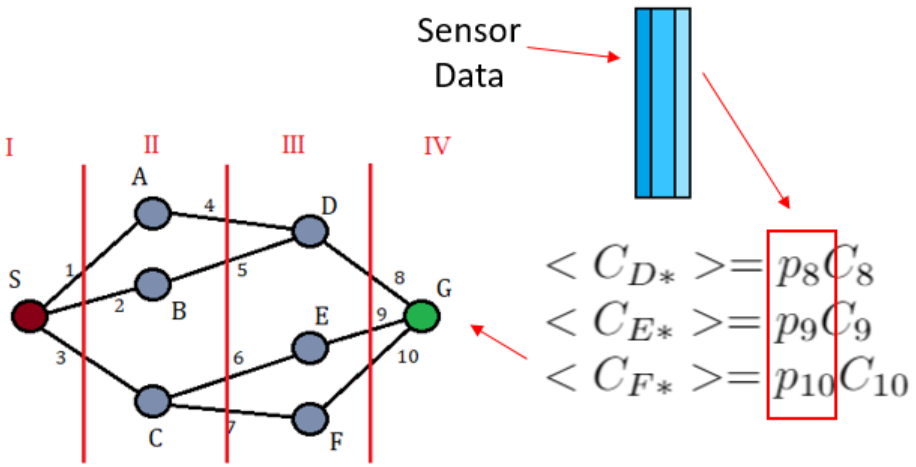

Figure 7.1: Application of sensor data as the means to adjust the conditional probabilities for implementing re-planning as a strategy for dynamic adaptive autonomy level calculation

The advantage, then, comes from the coupling of the VAP+ algorithm with some kind of well-established predictive system, of which there are many extant examples, to generate probability values across differing modules from sensor data. In this paradigm, the input data is constantly being pushed through the predictor to generate up to date conditional probabilities, which are incorporated into the re-planning phase of the VAP+ algorithm to consistently identify the optimal level of autonomy.

For systems such as this, the key innovation is decoupling prediction of A_L from the machine learning problem, which tends to be very difficult, given the already subjective nature of most autonomy level assignment systems and selection methods. Aside from this data based complication, there is also the problem of the inherent difficulty in learning such an abstracted problem. Under the VAP+ algorithm, the complex task of identifying A_L is removed from the domain of learning systems and the operator's tasks by leveraging properties of our definition, allowing for the integration of these predictive systems to tasks they are well established in solving to achieve a genuinely dynamic adaptable autonomy planner with rigorous level assignment and autonomous selection of the ideal operating level.

An additional benefit we can derive from machine learning based techniques, as highlighted in Figure 7.2, specifically, is the identification of failure modes from the data. By

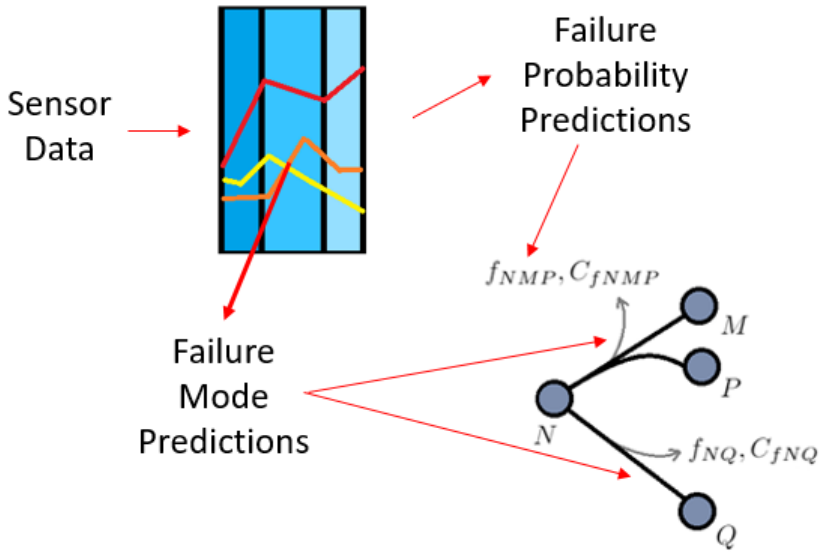

Figure 7.2: Use of machine learning for failure mode analysis to both select mixed mode failure cost estimates and determine failure mode grouping.

analyzing the performance of machine learning predictors when acting in failure cases, certain patterns in statistical analysis can allow the identification of systematic differentiation between modes with different failure costs and probabilities, ideally leading to responses which minimize error through planning effort.

REFERENCES

[1] "Rii track-1: Kentucky advanced manufacturing partnership for enhanced robotics and structures," https://www.nsf.gov/awardsearch/showAward?AWD_ID=1849213, Accessed: 8-10-21.

[2] "Fw-htf-rm: Enhancing future work of nursing professionals through collaborative human-robot interfaces," https://nsf.gov/awardsearch/showAward?AWD_ID=2026584, Accessed: 8-10-21.

[3] Angeliki Zacharaki, Ioannis Kostavelis, Antonios Gasteratos, and Ioannis Dokas, "Safety bounds in human robot interaction: a survey," *Safety science*, vol. 127, pp. 104667, 2020.

[4] Matthew Ball and Vic Callaghan, "Explorations of autonomy: an investigation of adjustable autonomy in intelligent environments," in *2012 Eighth International Conference on Intelligent Environments*. IEEE, 2012, pp. 114–121.

[5] Thomas B Sheridan, "Human–robot interaction: status and challenges," *Human factors*, vol. 58, no. 4, pp. 525–532, 2016.

[6] Shu Jiang and Ronald C Arkin, "Mixed-initiative human-robot interaction: definition, taxonomy, and survey," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2015, pp. 954–961.

[7] Robert J Anderson, "Autonomous, teleoperated, and shared control of robot systems," in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 1996, vol. 3, pp. 2025–2032.

[8] Bernard P Zeigler, "High autonomy systems: concepts and models," in *Proceedings [1990]. AI, Simulation and Planning in High Autonomy Systems*. IEEE, 1990, pp. 2–7.

[9] Krishnanand N Kaipa, Akshaya S Kankanhalli-Nagendra, Nithyananda B Kumbla, Shaurya Shriyam, Srudeep Somnaath Thevendria-Karthic, Jeremy A Marvel, and Satyandra K Gupta, "Enhancing robotic unstructured bin-picking performance by enabling remote human interventions in challenging perception scenarios," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2016, pp. 639–645.

[10] Christina Rödel, Susanne Stadler, Alexander Meschtscherjakov, and Manfred Tscheligi, "Towards autonomous cars: The effect of autonomy levels on acceptance and user experience," in *Proceedings of the 6th international conference on automotive user interfaces and interactive vehicular applications*, 2014, pp. 1–8.

[11] Salama A Mostafa, Mohd Sharifuddin Ahmad, and Aida Mustapha, "Adjustable autonomy: a systematic literature review," *Artificial Intelligence Review*, vol. 51, no. 2, pp. 149–186, 2019.

[12] Michael T Wolf, Christopher Assad, Matthew T Vernacchia, Joshua Fromm, and Henna L Jethani, "Gesture-based robot control with variable autonomy from the jpl biosleeve," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 1160–1165.

[13] C Miller, R Goldman, H Funk, Peggy Wu, and B Pate, "A playbook approach to variable autonomy control: Application for control of multiple, heterogeneous unmanned air vehicles," in *Proceedings of FORUM 60, the Annual Meeting of the American Helicopter Society*. American Helicopter Society International, Inc Baltimore, 2004, vol. 3, pp. 2146–2157.

[14] Paul M Fitts, "The information capacity of the human motor system in controlling the amplitude of movement.," *Journal of experimental psychology*, vol. 47, no. 6, pp. 381, 1954.

[15] M Bernardine Dias, Balajee Kannan, Brett Browning, E Jones, Brenna Argall, M Fred-

die Dias, Marc Zinck, M Veloso, and Anthony Stentz, "Sliding autonomy for peer-to-peer human-robot teams," in *Proceedings of the international conference on intelligent autonomous systems*, 2008, pp. 332–341.

[16] Sebastian Muszynski, Jörg Stückler, and Sven Behnke, "Adjustable autonomy for mobile teleoperation of personal service robots," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 933–940.

[17] Ryan C Johnson, Kristin N Saboe, Matthew S Prewett, Michael D Coovert, and Linda R Elliott, "Autonomy and automation reliability in human-robot interaction: A qualitative review," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. SAGE Publications Sage CA: Los Angeles, CA, 2009, vol. 53, pp. 1398–1402.

[18] Agostino De Santis, Bruno Siciliano, Alessandro De Luca, and Antonio Bicchi, "An atlas of physical human–robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[19] Andrea Cherubini, Robin Passama, André Crosnier, Antoine Lasnier, and Philippe Fraisse, "Collaborative manufacturing with physical human–robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 40, pp. 1–13, 2016.

[20] Julie A Adams, "Critical considerations for human-robot interface development," in *Proceedings of 2002 AAAI Fall Symposium*, 2002, pp. 1–8.

[21] Monika Jain, Ashwani Lohiya Aditi, Mohammad Fahad Khan, and Abhishek Maurya, "Wireless gesture control robot: an analysis," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, no. 10, pp. 855–857, 2012.

[22] Xianjie Pu, Hengyu Guo, Qian Tang, Jie Chen, Li Feng, Guanlin Liu, Xue Wang, Yi Xi, Chenguo Hu, and Zhong Lin Wang, "Rotation sensing and gesture control of a

robot joint via triboelectric quantization sensor," *Nano Energy*, vol. 54, pp. 453–460, 2018.

[23] Ed Colgate, Antonio Bicchi, Michael Aaron Peshkin, and James Edward Colgate, "Safety for physical human-robot interaction," in *Springer handbook of robotics*, pp. 1335–1348. Springer, 2008.

[24] Sami Haddadin, Alin Albu-Schaffer, Alessandro De Luca, and Gerd Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3356–3363.

[25] Christian J Bell, Pradeep Shenoy, Rawichote Chalodhorn, and Rajesh PN Rao, "Control of a humanoid robot by a noninvasive brain–computer interface in humans," *Journal of neural engineering*, vol. 5, no. 2, pp. 214, 2008.

[26] Narendra Prataksita, Yi-Tseng Lin, Hung-Chyun Chou, and Chung-Hsien Kuo, "Brain-robot control interface: Development and application," in *2014 IEEE International Symposium on Bioelectronics and Bioinformatics (IEEE ISBB 2014)*. IEEE, 2014, pp. 1–4.

[27] Sven Cremer, Fahad Mirza, Yathartha Tuladhar, Rommel Alonzo, Anthony Hingeley, and Dan O Popa, "Investigation of human-robot interface performance in household environments," in *Sensors for Next-Generation Robotics III*. International Society for Optics and Photonics, 2016, vol. 9859, p. 985904.

[28] R Grieder, J Alonso-Mora, C Bloechlinger, R Siegwart, and P Beardsley, "Multi-robot control and interaction with a hand-held tablet," in *ICRA 2014 Workshop on Multiple Robot Systems. IEEE*. Citeseer, 2014.

[29] Yan Liu, Cheng Chen, and Max Meng, "A study on the teleoperation of robot systems via www," in *2000 Canadian Conference on Electrical and Computer Engineering.*

Conference Proceedings. Navigating to a New Era (Cat. No. 00TH8492). IEEE, 2000, vol. 2, pp. 836–840.

[30] K Matsumaru, A Fhjimori, T Kotoku, and Kiyoshi Komoriya, "Action strategy for remote operation of mobile robot in human coexistence environment," in *2000 26th Annual Conference of the IEEE Industrial Electronics Society. IECON 2000. 2000 IEEE International Conference on Industrial Electronics, Control and Instrumentation. 21st Century Technologies*. IEEE, 2000, vol. 1, pp. 1–6.

[31] Martin Urban and Peter Bajcsy, "Fusion of voice, gesture, and human-computer interface controls for remotely operated robot," in *2005 7th International Conference on Information Fusion*. IEEE, 2005, vol. 2, pp. 8–pp.

[32] Jimmy Baraglia, Maya Cakmak, Yukie Nagai, Rajesh Rao, and Minoru Asada, "Initiative in robot assistance during collaborative task execution," in *2016 11th ACM/IEEE international conference on human-robot interaction (HRI)*. IEEE, 2016, pp. 67–74.

[33] Michael A Goodrich, Dan R Olsen, Jacob W Crandall, and Thomas J Palmer, "Experiments in adjustable autonomy," in *Proceedings of IJCAI Workshop on autonomy, delegation and control: interacting with intelligent agents*. Seattle, WA, 2001, pp. 1624–1629.

[34] S Lichiardopol, "A survey on teleoperation," *Technische Universitat Eindhoven, DCT report*, vol. 20, pp. 40–60, 2007.

[35] Carolina Passenberg, Angelika Peer, and Martin Buss, "A survey of environment-, operator-, and task-adapted controllers for teleoperation systems," *Mechatronics*, vol. 20, no. 7, pp. 787–801, 2010.

[36] David A Abbink, Tom Carlson, Mark Mulder, Joost CF de Winter, Farzad Aminravan, Tricia L Gibo, and Erwin R Boer, "A topology of shared control systemsfinding common ground in diversity," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 5, pp. 509–525, 2018.

[37] David Kortenkamp, R Peter Bonasso, Dan Ryan, and Debbie Schreckenghost, "Traded control with autonomous robots as mixed initiative interaction," in *AAAI Symposium on Mixed Initiative Interaction*, 1997, pp. 89–94.

[38] David Vernon, "Enaction as a conceptual framework for developmental cognitive robotics," *Paladyn, Journal of Behavioral Robotics*, vol. 1, no. 2, pp. 89–98, 2010.

[39] Rehj Cantrell, Matthias Scheutz, Paul Schermerhorn, and Xuan Wu, "Robust spoken instruction understanding for hri," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2010, pp. 275–282.

[40] Frederik W Heger, Laura M Hiatt, Brennan Sellner, Reid Simmons, and Sanjiv Singh, "Results in sliding autonomy for multi-robot spatial assembly," 2005.

[41] Manolis Chiou, Goda Bieksaite, Nick Hawes, and Rustam Stolkin, "Human-initiative variable autonomy: An experimental analysis of the interactions between a human operator and a remotely operated mobile robot which also possesses autonomous capabilities," in *2016 AAAI Fall Symposium Series*, 2016.

[42] Chin-Yin Huang and Shimon Y Nof, "Autonomy and viability-measures for agent-based manufacturing systems," *International Journal of Production Research*, vol. 38, no. 17, pp. 4129–4148, 2000.

[43] Changjoo Nam, Phillip Walker, Huao Li, Michael Lewis, and Katia Sycara, "Models of trust in human control of swarms with varied levels of autonomy," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 3, pp. 194–204, 2019.

[44] Michael R Benjamin, Henrik Schmidt, Paul M Newman, and John J Leonard, "Nested autonomy for unmanned marine vehicles with moos-ivp," *Journal of Field Robotics*, vol. 27, no. 6, pp. 834–875, 2010.

[45] Harold L Alexander, "Experiments in teleoperator and autonomous control of space robotic vehicles," in *1991 American Control Conference*. IEEE, 1991, pp. 1474–1477.

[46] Morgan Quigley, Michael A Goodrich, and Randal W Beard, "Semi-autonomous human-uav interfaces for fixed-wing mini-uavs," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. IEEE, 2004, vol. 3, pp. 2457–2462.

[47] Jenay M Beer, Arthur D Fisk, and Wendy A Rogers, "Toward a framework for levels of robot autonomy in human-robot interaction," *Journal of human-robot interaction*, vol. 3, no. 2, pp. 74, 2014.

[48] David J Bruemmer, Donald D Dudenhoeffer, and Julie L Marble, "Dynamic-autonomy for urban search and rescue.," in *AAAI mobile robot competition*, 2002, pp. 33–37.

[49] Janice A Klein, "A reexamination of autonomy in light of new manufacturing practices," *Human Relations*, vol. 44, no. 1, pp. 21–38, 1991.

[50] Cliff Joslyn, "Models, controls, and levels of semiotic autonomy," in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC) held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA) Intell.* IEEE, 1998, pp. 747–752.

[51] Fang Tang and Ethan Ito, "Human-assisted navigation through sliding autonomy," in *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2017, pp. 26–30.

[52] Jacob W Crandall and Michael A Goodrich, "Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation," in *IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2002, vol. 2, pp. 1290–1295.

[53] Dae-Jin Kim, Rebekah Hazlett-Knudsen, Heather Culver-Godfrey, Greta Rucks, Tara Cunningham, David Portee, John Bricout, Zhao Wang, and Aman Behal, "How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42, no. 1, pp. 2–14, 2011.

[54] Aaron Steinfeld, Terrence Fong, David Kaber, Michael Lewis, Jean Scholtz, Alan Schultz, and Michael Goodrich, "Common metrics for human-robot interaction," in *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 33–40.

[55] KS Barber and CE Martin, "Agent autonomy: Specification, measurement, and dynamic adjustment," in *Proceedings of the autonomy control software workshop at autonomous agents*. Citeseer, 1999, vol. 1999, pp. 8–15.

[56] Lu Feng, Clemens Wiltsche, Laura Humphrey, and Ufuk Topcu, "Synthesis of human-in-the-loop control protocols for autonomous systems," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 450–462, 2016.

[57] Krishnanand N Kaipa, Srudeep Somnaath Thevendria-Karthic, Shaurya Shriyam, Ariyan M Kabir, Joshua D Langsfeld, and Satyandra K Gupta, "Resolving automated perception system failures in bin-picking tasks using assistance from remote human operators," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015, pp. 1453–1458.

[58] Nicolas Côté, Arnaud Canu, Maroua Bouzid, and Abdel-Iillah Mouaddib, "Humans-robots sliding collaboration control in complex environments with adjustable autonomy," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. IEEE, 2012, vol. 2, pp. 146–153.

[59] Jim Mainprice, Calder Phillips-Grafflin, Halit Bener Suay, Nicholas Alunni, Daniel Lofaro, Dmitry Berenson, Sonia Chernova, Robert W Lindeman, and Paul Oh, "From autonomy to cooperative traded control of humanoid manipulation tasks with unreliable communication: System design and lessons learned," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3767–3774.

[60] Jie Fu and Ufuk Topcu, "Synthesis of shared autonomy policies with temporal logic

specifications," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 7–17, 2015.

[61] Giannis Petousakis, Manolis Chiou, Grigoris Nikolaou, and Rustam Stolkin, "Human operator cognitive availability aware mixed-initiative control," in *2020 IEEE International Conference on Human-Machine Systems (ICHMS)*. IEEE, 2020, pp. 1–4.

[62] Julie A Adams, Pramila Rani, and Nilanjan Sarkar, "Mixed initiative interaction and robotic systems," in *AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*. Citeseer, 2004, pp. 6–13.

[63] Rahul Chipalkatty, Greg Droge, and Magnus B Egerstedt, "Less is more: Mixed-initiative model-predictive control with human inputs," *IEEE Transactions on Robotics*, vol. 29, no. 3, pp. 695–703, 2013.

[64] Marc Rigter, Bruno Lacerda, and Nick Hawes, "A framework for learning from demonstration with minimal human effort," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2023–2030, 2020.

[65] A Hong, O Igharoro, Y Liu, Farzad Niroui, Goldie Nejat, and Beno Benhabib, "Investigating human-robot teams for learning-based semi-autonomous control in urban search and rescue environments," *Journal of Intelligent & Robotic Systems*, vol. 94, no. 3, pp. 669–686, 2019.

[66] T Chatchanayuenyong and M Parnichkun, "Neural network based-time optimal sliding mode control for an autonomous underwater robot," *Mechatronics*, vol. 16, no. 8, pp. 471–478, 2006.

[67] Jung Ju Choi, Yunkyung Kim, and Sonya S Kwak, "The autonomy levels and the human intervention levels of robots: The impact of robot types in human-robot interaction," in *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2014, pp. 1069–1074.

[68] F Montes-Gonzalez, TJ Prescott, and J Negrete-Martinez, "Minimizing human intervention in the development of basal ganglia-inspired robot control," *Applied Bionics and Biomechanics*, vol. 4, no. 3, pp. 101–109, 2007.

[69] Dong-Hyun Lee, Ji-Hyeong Han, and Jong-Hwan Kim, "A preference-based task allocation framework for multi-robot coordination," in *2011 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2011, pp. 2925–2930.

[70] Jim Mainprice, Mamoun Gharbi, Thierry Siméon, and Rachid Alami, "Sharing effort in planning human-robot handover tasks," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2012, pp. 764–770.

[71] JP Gunderson and WN Martin, "A probability–aware planning architecture to support variable autonomy," *American Association for Artificail Intelligence Technical Report SS-03*, vol. 4, pp. 83–88.

[72] Emmanuel Senft, Paul Baxter, James Kennedy, Séverin Lemaignan, and Tony Belpaeme, "Supervised autonomy for online learning in human-robot interaction," *Pattern Recognition Letters*, vol. 99, pp. 77–86, 2017.

[73] R Saravanan, S Ramabalan, N Godwin Raja Ebenezer, and R Natarajan, "Evolutionary bi-criteria optimum design of robots based on task specifications," *The International Journal of Advanced Manufacturing Technology*, vol. 41, no. 3-4, pp. 386–406, 2009.

[74] LJ Manso, P Bustos, R Alami, G Milliez, and P Núnez, "Planning human-robot interaction tasks using graph models," in *Proceedings of International Workshop on Recognition and Action for Scene Understanding (REACTS 2015)*, 2015, pp. 15–27.

[75] Panagiota Tsarouchi, Sotiris Makris, and George Chryssolouris, "Human–robot interaction review and challenges on task planning and programming," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 916–931, 2016.

[76] Manolis Chiou, Mohammed Talha, and Rustam Stolkinl, "Learning effects in variable autonomy human-robot systems: how much training is enough?," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 720–727.

[77] Luis Miguel Molina and Francisco Javier Llorens-Montes, "Autonomy and teamwork effect on knowledge transfer: knowledge transfer-ability as a moderator variable," *International journal of technology transfer and commercialisation*, vol. 5, no. 3, pp. 263–280, 2006.

[78] Manolis Chiou, Nick Hawes, Rustam Stolkin, Kimron L Shapiro, Jess R Kerlin, and Andrew Clouter, "Towards the principled study of variable autonomy in mobile robots," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2015, pp. 1053–1059.

[79] Christopher Robinson, Indika B Wijayasinghe, and Dan O Popa, "Quantitative variable autonomy levels for traded control in a pick-and-place task," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2019, pp. 697–702.

[80] Shamsudeen Abubakar, Sumit K Das, Chris Robinson, Mohammed N Saadatzi, M Cynthia Logsdon, Heather Mitchell, Diane Chlebowy, and Dan O Popa, "Arna, a service robot for nursing assistance: System overview and user acceptability," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1408–1414.

[81] Mohammad Nasser Saadatzi, M Cynthia Logsdon, Shamsudeen Abubakar, Sumit Das, Penelope Jankoski, Heather Mitchell, Diane Chlebowy, and Dan O Popa, "Acceptability of using a robotic nursing assistant in health care environments: Experimental pilot study," *Journal of Medical Internet Research*, vol. 22, no. 11, pp. e17509, 2020.

[82] Mohammad N Saadatzi, Shamsudeen Abubakar, Sumit Kumar Das, M Hossein Saa-

datzi, and Dan Popa, "Neuroadaptive controller for physical interaction with an omnidirectional mobile nurse assistant robot," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2020, vol. 83990, p. V010T10A055.

APPENDIX A

LIST OF PUBLICATIONS

1. Robinson, Christopher, Indika B. Wijayasinghe, and Dan O. Popa. "Quantitative Variable Autonomy Levels for Traded Control in a Pick-and-Place Task." 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). IEEE, 2019.

2. Robinson, Christopher, Mohammad N. Saadatzi, and Dan O. Popa. "Bin-Picking using Model-Free Visual Heuristics and Grasp-Constrained Imaging." 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). IEEE, 2019.

3. Robinson, Christopher, Shamsudeen Abubakar, Sumit Kumar Das, and Dan O. Popa. A High-modularity ROS to Firmware Instrumentation Bridge for Robotic Sensors 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE). IEEE, 2020.

4. Cassady, Jacob T., Chris Robinson, and Dan O. Popa. "Increasing user trust in a fetching robot using explainable AI in a traded control paradigm."Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments. 2020.

5. Aneseh Alvanpour, Sumit Kumar Das, Christopher Robinson, Olfa Nasraoui, Dan O. Popa. Robot Failure Mode Prediction with Explainable Machine Learning 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE). IEEE, 2020.

6. Robinson, Christopher, Shamsudeen Abubakar, Sumit Kumar Das, and Dan O. Popa.

"Variable Autonomy Level Assignment for Human Robot Interaction Tasks" IEEE Transactions on Automation Science and Engineering (Under Review)

7. Robinson, Christopher, Shamsudeen Abubakar, Sumit Kumar Das, and Dan O. Popa. "Evaluation of Variable Autonomy Assignment in a Bin Picking Task" IEEE Transactions on Human Machine Systems (Under Review)

CURRICULUM VITAE

Christopher Robinson

6000 Noah Drive

Louisville, KY 40258

(502)544-5023

ckevinr@gmail.com

PhD, Electrical and Computer Engineering

University of Louisville Speed School of Engineering, Louisville KY, 2017-21

M.S., Electrical and Computer Engineering

University of Louisville, Speed School of Engineering, Louisville KY, 2012-14

B.S., Mathematics / Physics

Centre College, Danville KY, 2008-12

Experience:

U of L Next Gen Systems Lab; Research Assistant; 8/16 - Present

- Perform research investigating the application of Machine Learning and Artificial Intelligence tasks to implementation of effective robotic control systems in the context of human-robot interaction systems, with a focus on applying AI and Machine learning to Human/Robot interaction and Autonomous Manipulation

- Develop software for both implementation and research cases of topics relating to scientific programs within the lab, including Sensorization, Automated Planning, Computer Vision, Machine Learning, and Autonomous Manipulation assets

- Management of robotics laboratory spaces, including purchasing, organization, equipment maintenance, and inventory management for supplies and research articles.

- Directing undergraduate and master's students in performance of research and development-related tasks.

U of L Conn Center for Renewable Energy; Research Assistant; 8/16 12/17

- Support Conn Center research initiatives by assisting with the design, development, assembly and operation of robotic equipment for chemical experiments in automated catalyst production.

- Design robotic systems for liquid handling and deposition processes, and for implementation of in-situ plasma processing of liquid feedstock.

- Write control and interface software for operating custom laboratory equipment and generation of test assay files for experiments.

- Contribute electronic and electrical engineering expertise to the addressing of research questions regarding systems implementation, testing, and evaluation for chemical engineering processes.

- Develop and analyze algorithmic approaches to solving practical laboratory operations problems, in particular those involving the preservation of valuable chemical assets.

Durham Labs; Lead Engineer; 2/14 - 7/16

- Design electronic and robotic articles for product development, including environmental sensing apparatus, educational development tools and autonomous robots.

- Write control, interface and data processing software supporting electronic products and internal projects, including analysis and processing packages, web scrapers, embedded controllers and task automation applications.

- Prepare prototypes for product testing and evaluation.

- Analyze and implement development and production processes for peak efficiency and high quality.

- Analyze and troubleshoot bugs throughout product development and resolve technical issues which cause them.

- Provide technical consulting for marketing, planning, and educational materials.

- Develop and execute test protocols for products and internal tools.

- Design and layout of electronic PCBs.

- Preparation of technical documentation pertaining to projects.

- Provide technical oversight for subordinate engineers.

Centre College Physics Dept.; Research Assistant; 5/09-8/10, 5/10-8/11

- Read and evaluate scientific papers related to our field of study and analyze mathematical models for applicability to simulation tasks.

- Develop software to perform simulation tasks and implement mathematical models from relevant prior work, as well as additional modeling efforts from research.

- Investigate approaches to improve simulation computational efficiency and establish the validity of these efforts by mathematical evaluation and comparison to established results.

- Create software resources based on computer vision methodology to enable flexible large-scale testing of simulation parameters.

- Develop algorithms for quick processing, evaluation and categorization of large data sets comprising batch outputs from simulations.

Publications:

1) Robinson, Christopher, Indika B. Wijayasinghe, and Dan O. Popa. QuantitativeVariable Autonomy Levels for Traded Control in a Pick-and-Place Task. 2019 IEEE15th International Conference on Automation Science and Engineering (CASE). IEEE,2019.2.

2) Robinson, Christopher, Mohammad N. Saadatzi, and Dan O. Popa. Bin-Pickingusing Model-Free Visual Heuristics and Grasp-Constrained Imaging. 2019 IEEE 15thInternational Conference on Automation Science and Engineering (CASE). IEEE,2019.3.

3) Robinson, Christopher, Shamsudeen Abubakar, Sumit Kumar Das, and Dan O. Popa.A High-modularity ROS to Firmware Instrumentation Bridge for Robotic Sensors 2020IEEE 16th International Conference on Automation Science and Engineering (CASE).IEEE, 2020.4.

4) Cassady, Jacob T., Chris Robinson, and Dan O. Popa. Increasing user trust in afetching robot using explainable AI in a traded control paradigm.Proceedings of the13th ACM International Conference on PErvasive Technologies Related to AssistiveEnvironments. 2020.5.

5) Aneseh Alvanpour, Sumit Kumar Das, Christopher Robinson, Olfa Nasraoui, DanO. Popa. Robot Failure Mode Prediction with Explainable Machine Learning 2020IEEE 16th International Conference on Automation Science and Engineering (CASE).IEEE, 2020.6.

6) Robinson, Christopher. "A vehicle routing heuristic: Derived from altered problem constraints." 2017 Computing Conference. IEEE, 2017.

7) Robinson, Christopher "Modified reinforcement learning for sequential action behaviors and its application to robotics." IEEE SOUTHEASTCON 2014. IEEE, 2014.

8) Robinson, Christopher, Shamsudeen Abubakar, Sumit Kumar Das, and Dan O. Popa.157 Variable Autonomy Level Assignment for Human Robot Interaction Tasks IEEETransactions on Automation Science and Engineering (Under Review)

9) Robinson, Christopher, Shamsudeen Abubakar, Sumit Kumar Das, and Dan O. Popa.Evaluation of Variable Autonomy Assignment in a Bin Picking Task IEEE Trans-actions on Human Machine Systems (Under Review)

Awards:

- ICRA 2018 Tidy My Room Challenge: Sole qualifying contender
- Maker of Merit Award, Louisville Mini Maker Faire 2015, 9/15
- Speed School Mechatronics Robotics Competition: 'Treasure Hunt' Winner; 4/14
- The Marshall Wilt Physics Prize, Centre College, 6/12
- Centre College Olin 500 autonomous robotics competition: First Place, 6/12
- Nerdkits.com Featured Project Computer Controlled Robotic Arm, 7/10

Invited Presentations:

- TJUC Invited Lecture: Strange Bedfellows; Science and Religion in a Universe too big for one Mirror 6/17
- RE3 Conference on Renewable Energy; Poster presentation: Automated Plasma Synthesis for Catalyst Production 5/16
- Centre College Physics Dept.; Invited speaker for Sigma Pi Sigma, 10/15
- U of L Graduate Research Symposium, Oral Presentation (Independent Research: Competitive Inhibition of Biological Agents), 3/13
- University of Miami, Ohio Annual Mathematics Conference, 10/09; Presentation (Research: Modeling Snow Crystal Growth with Cellular Automata)