Autonomous Parking

Unlocking Value for Real Estate Developers Using Robotics, Software & Machine Learning

Volley Automation, Inc.

February 1, 2019

Contents

List of Figures

List of Tables

Nomenclature

AGV Automated Guided Vehicle

COTS Commercial Off-The-Shelf

MADM Multi-Attribute Decision Making

MCDM Multi-Criteria Decision Making

MTBF Mean Time Between Failure

MTTR Mean Time to Repair

NSGA-II Non-dominated Sorting Genetic Algorithm-II

TOPSIS Technique for Ordered Preference Selection by Similarity to Ideal Solution

Abstract

This report provides a comprehensive review of the autonomous parking solution proposed for implementation by Volley Automation. The report begins by defining the challenge along with the various stakeholders. In essence, there are 2 primary stakeholders: 1) the end users and 2) the garage owner/building developer. These stakeholders drive the primary metrics of interest: 1) Minimize customer wait time, 2) Maximize space efficiency and/or minimize required garage size, 3) Maximize throughput, and 4) Minimize operational cost.

With respect to these metrics, the stakeholders are clearly at odds. In general, the users want their car as quickly as possible while the developers wish to reduce space and costs while meeting the minimum throughput requirements. These conflicting desires represent the gap that a properly designed autonomous parking *solution* can bridge. To avoid pitfalls that have plagued previous systems, the proposed solution looks to encompass and integrate all disciplines related to the problem including the garage layout design, robotic system, electromechanical systems, planning and controlling, fault tolerance, and user experience. Throughout this report, the novel software and hardware concepts developed by Volley are presented. Further, many of the previous failures have occurred due to poor integration between these disciplines, so by better planning for and testing against the integration challenges, the proposed solution represents a viable path forward for both the end user and developer.

Chapter 1

Introduction

As more people move to cities with ever-increasing density, the burden of parking vehicles and storing payloads is beginning to overwhelm traditional parking structures. In light of these developments, new solutions that address parking needs more efficiently and can serve additional transportation modes and customer services is required. Automated parking garages can work to address these needs by increasing parking density and presenting a well-controlled area to provide services. This work presents Volley Automation's proposed design to provide a robust, automated parking solution that addresses the shortcomings of previous attempts while adding features that fit into the current smart-phone, services ecosystem.

This chapter begins by discussing the parking industry in general and the role that autonomous parking can play in the industry. Gaps in current systems and solutions are then identified based on observations from previous autonomous parking attempts. Finally, the components of an autonomous parking system are discussed alongside the utility functions to optimize when designing the autonomous parking solution.

1.1 Parking Landscape Overview

An autonomous parking system can enable improved parking options by increasing the car density in a parking structure while eliminating the need for customers to drive around to find a space. This is, therefore, a win-win concept that can reduce parking structure construction costs, enable smaller structures to support the required number of cars, and improve the customers' experience.

Despite the potential benefits of such a system, autonomous parking is not prevalent in today's world. The technology for this type of system is clearly available... self-driving cars are being driven thousands of miles each day, elevators are extremely reliable, and automated guided vehicles (AGVs) pick items from warehouses across the world. Integration and implementation gaps, therefore, must exist that are preventing more widespread adoption. The remainder of this section discusses the categories of parking locations and uses, types of parking solutions, and where opportunities exist to bridge the gaps that have plagued previous companies.

1.1.1 Categories of Parking

Many parking solutions exist from open lots to multi-floor garages to autonomous systems. Before discussing these solutions, this section focuses on categorizing parking locations and uses to ensure the proper solution is recommended. Ultimately, while Volley provides automated parking solutions, this solution is not appropriate for all situations, so an understanding of the parking landscape is necessary to provide the best recommendations to our customers. Further, the service level and density requirements of the automated solution will vary based on the use of the space, payment strategy, and customer expectations.

The main factors that influence the selection of a parking solution are the peak-hour volume (the number of entrances and exits from the structure) and the total number of spaces required. The number of spaces required (examples provided in Table 1.1) is typically dictated by regulations [5, 6]. Regulations also commonly require that the structure can handle some percentile of the peak-hour volume (85–90%) [7].

Table 1.1: Example Parking Space Requirements [1, 2]

Primary Use	Parking Space Requirement	Peak Demand Time
Multi-family Housing	$^{1.5}/_{\text{unit}}$	Evening
Hotel	$1 - {}^{1.25}/_{\text{room}}$	Weekday evening
Hospital	$^{0.4}/_{\text{Employee}} + {}^{1}/_{3}$ Beds+ $^{1}/_{5}$ Outpatients + $^{1}/_{4}$ Medical Staff[1]	Weekday-day
General Retail	$^{3.3}/_{1000\text{ sq.ft.}}$	Saturday-day
Office Building	$^{3.6}/_{1000\text{ sq.ft.}}$	Weekday-day

Based on the parking requirements derived from calculations similar to those presented in Table 1.1 and the availability and cost of land, the developer chooses an appropriate parking solution. The following section reviews the parking solution options and their relative costs.

1.1.2 Survey of Parking Solutions

The parking landscape contains a wide variety of solutions that support different demands, types of parkers (e.g. prepaid, long-term, hourly, validated, etc.), land availability and cost, and customer expectations. These solutions range from surface lots (cheapest but require the most land) to fully autonomous structures (traditionally most expensive and risky, least amount of land). Within this range are a number of solutions that contain different garage layouts, additional stories, and varied payment and access controls that can help tailor the solution to a specific need.

Ultimately, the developer's goal is to satisfy the required parking demand for the lowest cost (whether that be actual construction and operating costs or opportunity cost to dedicating more land or stories for parking). As such, the primary driver for selecting a parking solution is to compare the area available for parking with the number of spaces required. Then depending on the parking solution chosen, the cost per space can range from \$10,000–\$25,000/stall [3–5, 8]. Table 1.2 presents very roughly estimated costs for various types of conventional and automated parking solutions. Generally, the data indicates that for a stand-alone parking structure, a conventional parking structure is cheaper and not all that less space efficient than the corresponding automated solution. The efficiency of these solutions could also be improved by leveraging car stackers. However, for parking that is integrated with a building, the automated solution is estimated to double the space efficiency

and reduce the cost per space. Hence, automated parking solutions should be first leveraged in places where space is at a premium and/or in cases where the parking is integrated with the building.

Table 1.2: Parking Structure Cost and Efficiency Comparison (Reproduced from [3, 4])

Configuration	Type	Unit Cost/ Sq. Ft.	Efficiency Sq. Ft./ Stall	Building Cost per Stall	Automated Parking Equipment Cost	Total Cost per Stall
Stand-alone, above grade	Conventional	$50	320	$16,000	$0	$16,000
	Automated	$45	225	$10,125	$16,000	$26,125
Below building, above grade	Conventional	$75	450	$33,750	$0	$33,750
	Automated	$65	225	$14,625	$16,000	$30,625
Below building, below grade	Conventional	$105	450	$47,250	$0	$47,250
	Automated	$85	225	$19,125	$16,000	$35,125

Two primary categories of automated parking solutions exist: 1) Rack-and-Rail systems and 2) AGV systems. The rack-and-rail system uses steel racks to store vehicles. The vehicles are moved to the racks via lifts and shuttles along a rail system. AGV systems instead use AGVs and lifts to move vehicles to determined spaces in the garage. Generally, the rack-and-rail systems are suited for very regularly shaped structures. In general, these systems use specially built steel rail systems to hold and move the cars without a concrete structure. The fixed steel construction limits opportunities to reconfigure the space, accommodate non-regularly shaped structures, respond to system malfunctions, and ultimately, reclaim the space if parking requirements decrease in the future.

The AGV systems leverage a more conventional concrete structure to house the parked payloads that more easily comply with fire safety regulations and are easier to maintain. Further, the concrete structure can be re-purposed if the demand for parking decreases. The maneuverability of the AGV also allows the system to leverage irregularly shaped structures to park and retrieve cars, which is important to ensure peak utilization of the space. The AGV paradigm ultimately offers the ability to get the highest parking density of all parking solutions and is the primary solution pursued by Volley. The following section discusses the general design process that Volley is leveraging to identify the system to implement.

1.2 Design Process

Automated parking is a field fraught with failures and mistrust. As such, new entrants must demonstrate that their solution addresses previous failures and is logically though out. The top-down design decision support process is a commonly applied framework to engineering design problems that helps to ensure logical and traceable decisions are made throughout [9]. The process is as follows:

Establish Need: Identify the overall need for the system being designed and its key characteristics.

Define the Problem: Research and define challenging problems in the design for the present or similar systems.

Establish Value: Define the metrics against which a design alternative is measured.

Generate Feasible Alternatives: Brainstorming and initial design step to create options to evaluate against the metrics.

Evaluate Alternatives: Setup a testing and simulation environment to evaluate the alternative concepts.

Make Decisions: With the concepts evaluated, make a decision about the concept to move forward with.

This process is followed throughout the paper to justify the design decisions made for the system at hand. The need for autonomous parking is well-established; cities are continuing to increase population density, thereby requiring more space for people and leaving less for parking. Previous attempts to satisfy this need have failed, so the following section identifies the gaps in previous solutions to help inform the problems that Volley must rectify.

1.3 Gap Analysis & Problem Definition

As cities continue to build vertically and increase their population density, the relative area available for parking is continuing to decrease. This has led to the need for automated parking solutions in general and AGV-based systems in particular. Despite the clear desire for this system among developers and users [10], autonomous garages have been fraught with problems [11, 12]. Through discussions around the industry, Volley Automation has identified 4 gaps in current implementations to bridge:

1. Software-based planning and routing algorithm deficiencies
2. Excessive complexity, cost, and customization of mechanical components
3. Limited consideration for user experience or customer interactions
4. Poor systems-level design and analysis

These gaps are discussed in the remainder of this section. Chapter 2 discusses Volley's proposed solutions to bridge the identified gaps.

1.3.1 Routing Deficiencies

Previously attempted automated parking systems performed well under nominal, low-stress conditions. However, during peak hours or when there was a hardware failure, the systems degraded quickly. In fact, at the Brickell House garage, residents were waiting up to an hour during peak weekend hours to retrieve their cars [11]. In this case, Boomerang had outsourced the control software to JBT, which primarily focuses on developing software for the control of AGVs in warehouses. On the surface, this seems like an adjacent problem; however, there are key differences between warehouse and parking planning.

The implemented software solutions rely on static routes that are pre-computed to insert and retrieve vehicles from the parking garage. This is an extremely robust solution for warehousing where there are fully dedicated, one-way aisles for moving goods. However, if a developer has resorted to autonomous parking, then there is necessarily a limited amount of space available for parking. This means that 2-way aisles, which naturally prevent gridlock, are not viable. There are also no longer alternative routes that are reliably available in the case of a failure somewhere else on the grid. Warehouse solutions also know well in advance

of retrieval requests and can thus better plan and schedule the movement. In the case of a parking garage when the expected lead time is between 2–10 minutes, flexibility in the planning system is key to driving down this lead time. In addition to Boomerang Systems, Park Plus uses statically calculated routes for their small number of automated parking garages.

On the other hand, other solutions have attempted to use an opportunistic A^* algorithm to route their retrievals. The A^* and similar algorithms route vehicles closer to their destination even if there is no clear path currently available. This works very well for systems with many paths to a destination (e.g. routing a trip with Google Maps), however with a very limited number of paths in a parking structure, the opportunistic algorithm can lead to gridlock very easily.

Volley seeks to overcome these deficiencies by implementing a routing system that calculates paths on the fly when an insertion or retrieval is triggered. The path network in a parking structure is relatively small, so Dykstra's algorithm can very quickly find the weighted shortest path from a source to destination. The base algorithm can be modified to identify the best route out from a blocked position, account for the actual time to accelerate and decelerate the AGV, and accommodate other robustness considerations (e.g. not blocking a lift). Then, by locking the path, gridlock cannot occur while still allowing parallel jobs that can dynamically find another non-intersecting path to their destination. Ultimately, this is a more conservative approach than the A^* implementation, but it avoids gridlock problems while improving on the time and robustness of static routing.

1.3.2 Mechanical Design Complexity

When designing a mechanical system, reducing the design complexity, limiting custom components, and reducing part counts and assembly steps are critical principles to reduce the system's cost and improve its "-ilities" (e.g. producibility, serviceability, reliability, etc.) [13–15]. Cost reductions are important for any system; however, especially for an automated parking system, where the quantity and reliability of the mechanical components can significantly impact the system's robustness, reducing costs to increase the quantity of AGVs, lifts, and bays is paramount.

The AGVs designed and built by Boomerang cost ~$120,000 to produce. This cost stemmed from high complexity and customization in each component. Most structural components required custom machining and lengthy assembly processes. The lifting mechanism is excessively complex due to challenging requirements on its height and flatness, which led to high cost and reliability challenges. Finally, the electrical systems within the AGV have many different voltage requirements, which leads to multiple circuits running through the system. Other companies, who appear "inspired" by Boomerang, also have similar deficiencies in their design.

Expanding beyond the component-level complexity, the overall locomotion concept is not ideal. As seen throughout this gap analysis, Boomerang appears to have started from a warehouse concept and adapted those strategies for automated parking. While this derivative design approach is valid, it tends to limit the creativity of designers. Warehouse robots, such as those employed by Amazon, commonly slide beneath shelving units, lift the shelves, and move them to where they need to go. As the shelves are relatively light, the robots are not that large and costly; however, when applied to lifting multiple ton vehicles, everything about the system becomes larger, more costly, and more difficult to maintain.

The current AGV system couples lifting the payload and moving the payload. This means that even though moving a heavy payload on wheels is relatively easy (e.g. it is possible to

push a car in neutral), the motors, brackets, and wheels must be sized to support the full weight of the car and tray. There are means, however, to decouple these functions to provide a simpler system. This can be accomplished by placing castors on the vehicle trays such that they can translate freely (when the brake is off). Then, the AGV itself only needs to provide enough force to push the wheeled tray, which can be accomplished by significantly smaller and less expensive motors. This concept is explored fully in Section 2.4.

1.3.3 User Experience

The explosion of app-based services in recent years demonstrates the value of selling a range of services to different customers. For example, Uber has many options ranging from economy, shared rides to luxury black cars, with accompanying price differences, available such that customers can pay different amounts for different services. Contrary to this, traditional parking structures charge a fixed price/rate for a generic parking space. Customers use price as an indicator of expected service level and can judge their experience based on the ratio of the perceived to expected level of service [16, 17]. Hence, as the service level is inherently different based on the spot that a car is parked, providing differentiated pricing can work to smooth the relationship between the service level and customer satisfaction. In the end, previous attempts at the automated parking concept have failed to explore price differentiation or other customer experience improvements that could manage customer expectations to increase satisfaction.

1.3.4 Systems Design Deficiencies

The final gap that is identified by Volley is deficiencies in the system design, evaluation, and testing practices of previous companies. At its core, an automated garage is a relatively simple system to simulate and analyze. Historical demand for inserts and retrievals can be gathered for any type of garage, and the actual simulation requires very little logic. Therefore, simulation should be a very powerful tool to determine whether the garage layout, equipment selection, and routing logic can accommodate the range of demand expected throughout the day.

By examining simulation reports from Boomerang Systems and general experience with AGV simulation consultants, there are a number of deficiencies identified. Firstly, both the Brickell House and Costa Hollywood garages had queuing and traffic flow studies conducted by an external consultant. In general, these types of capacity analyses should be conducted to ensure the system can handle times of high demand and stress. However, in both studies, many assumptions were made that reduced the demand on the system. These assumptions can lead to false confidence in the system's performance. Additional in house simulation studies from Boomerang also failed to investigate a variety of scenarios and stresses, thereby further contributing to the false expectations of performance.

Previous simulation studies also do not directly test the performance of the system controller code. Instead, the simulation builder codes logic into the simulation to mimic the controller. While this can help to complete a basic capacity analysis, the simulation cannot properly test and verify that the controller software can handle high demand situations and breakdowns. This means that the simulation demonstrates that the baked in logic can handle these situations, however there is no reason that the core controller's logic will preform similarly. As such, current practices miss a valuable opportunity to test the system in simulation.

The traditional project proposal and design process is also unsuitable for autonomous

parking. In a typical parking structure RFP process, the bidding company provides a single solution with n parking spots and a fixed construction cost. This solution will have the correct number of entry and exit lanes for the expected demand, and the rest of the parking process is up to the human driver (and any delays experienced by the human are blamed on other human drivers). With autonomous parking, any issues or delays are directly attributable to the garage (and the developer by extension). Therefore, the traditional process, which selects the minimum cost design that meets the parking requirements, is not sufficient to cover the truly multi-objective considerations involved in autonomous parking.

The final design process deficiency is a lack of integration testing before commercial implementation. This is likely a result of funding issues more than a strategic decision, but the first full implementation for most companies has been a customer garage. Any system with this amount of complexity and user interaction will have integration problems. Dedicated and thorough testing is necessary to work out these issues to ensure the first customer has a good experience. Especially with the previous high-profile failures in this industry, properly testing and demonstrating the technology will be paramount to building confidence in the product.

1.4 Chapter Summary

This chapter has introduced the overall parking landscape and where autonomous parking fits into the present and future of parking. Then gaps that have plagued previous attempts to bring this technology to market are discussed. These gaps help to drive Volley's design process presented in the following chapter.

Chapter 2

System Design

This chapter discusses Volley's autonomous parking structure design. The choices and decisions made by Volley are informed by the problems in current systems discussed in Section 1.3. This chapter begins by discussing a generic AGV-based autonomous parking system. Then, an overview of Volley's specific system is presented before detailing specific improvements and their implications.

2.1 System Overview

The AGV-based, autonomous parking system is comprised of the following elements:

- The parking structure itself with parking spaces and a customer pick-up/drop-off area
- The robotic means to transfer cars from the pick-up/drop-off point and the parking area
  - Automated guided vehicles
  - Automated lifts
  - Entry/exit bays
- The control structure that determines where cars are parked to maximize space efficiency and minimize customer wait time
- The user interface including pricing structure, insertion and retrieval planning policies, and penalty/reward structures

As these elements are all highly coupled, they should be designed concurrently to maximize the overall utility of the system. To establish the value for the system, the evaluation criteria include:

Minimize structure size/Maximize density: For an expected demand, reducing the size of the parking structure will reduce construction cost, maintenance cost, and allow the structure to fit in a reduced footprint. Conversely, increasing the density in a given space provides more spaces that can be sold to increase revenue.

Minimize customer wait time: Reducing the time that a customer must wait for his or her vehicle is essential. Ultimately, if the wait is too long and the request queue becomes unmanageable, customers will not return to the garage in the future. This is exacerbated by the fact that customers expect more from automated systems than from similar manual processes (e.g. a customer is likely willing to wait longer for a human valet than an automated parking system).

Maximize throughput: The throughput is essentially the number of cars per hour that can be processed by the garage. Increasing this is necessary to meet demand during peak hours of garage utilization.

Minimize operational cost: The operational cost covers the costs to operate the robots and garage systems along with any other maintenance and valet personnel. Minimizing the number of robots, lifts, and bays in the system will reduce the overall operational cost.

The remainder of this section describes the elements of the parking system in detail and discusses how each element maps to the evaluation criteria.

2.1.1 Parking Structure

Depending on the situation, the developer may be interested in getting the highest density for a given structure size or using the smallest structure to support a given number of parking spaces. Either way, the need for improved density beyond that available from a traditional parking garage drives the decision to pursue an automated parking solution. As such, it is imperative that the system takes advantage of as many space saving opportunities as possible without sacrificing a smooth end-user experience.

By using an AGV-driven parking solution, the parking density can be increased through the following means:

- Individual parking space sizes are reduced because the car no longer must angle into the space and the doors do not need to be opened
- Driving lanes and ramps are not needed
- Opportunity to reduce ceiling height
- No pedestrian accommodations (stairs, elevators, etc.)
- Cars may be double, triple, etc. parked because the robots can unblock the way for a car to be retrieved
- Non-regular spaces that could not be accessed by a vehicle required to turn become accessible
- Cars can be parked in different sized spots based on their dimensions... a Mini does not take the same space as an F-150

As the parking density is increased, the average user retrieval time will begin to worsen. This is because paths are more likely to be blocked by other parked cars or redundant paths in the form of extra bays, lift, and waypoints can be eliminated to provide additional parking spots. Some of these challenges can be mitigated by the static layout of the parking nodes

and the strategy to fill the garage. For instance, the parking spots can be selected such that cars block the fewest other cars when inserted into the garage.

The placement and number of lifts and bays are also important to the cost of the structure and the retrieval time. More lifts and bays increase the number of paths down to a customer, which will reduce resource conflicts. Further, the placement of the lifts and bays can increase the redundant paths in the garage, thereby reducing the impact of downtimes.

2.1.2 Electro-mechanical Systems

The primary electro-mechanical systems are the AGVs, lifts, and bays that automatically insert and retrieve cars for a customer. The performance of each clearly has a direct impact on the overall customer experience. Additionally, the cost of the systems must be considered to not only reduce the cost of the overall project but also allow the developer to deploy more AGVs, lifts, and/or bays to improve the overall system's robustness. Finally, the reliability and maintainability of each system is key to ensuring a high level of service. Generally, the system will perform well under nominal cases... the real challenge will be to recover from or prevent instances where the garage is highly occupied and one or many electro-mechanical systems go down.

Automated Guided Vehicles (AGVs)

The automated guided vehicles (AGVs) move customer cars or payloads between parking spaces and the customer bays. These vehicles are controlled by the central garage control system and move along a prescribed grid layout. The vehicles receive a set of instructions from the control system and execute these instructions with knowledge of the grid layout. As such, the vehicles are responsible for knowing and correcting their position and moving along straight lines sent from the controller. Otherwise, the vehicles contain a variety of safety sensors to correct their position, detect obstacles in their path, and provide redundant checks for lift position. The overall control system is designed such that the vehicles only require knowledge of the grid layout to control their own movement while providing hardware checks for obstacle and lift detection.

Lifts

The lifts move AGVs and/or payloads to different floors in the parking garage. Lifts are again controlled by the central control software and contain a mechanical switch/lock to confirm position on a particular floor. Only once the lift confirms it is secure will an AGV attempt to cross into or through a lift. Lifts may not necessarily be needed if a garage only has a single floor.

Bays

The bays are where customers deposit or retrieve their cars. Thus, this represents the boundary between the system and the outside world. As such, bays require extensive access controls to ensure that customers are not in the bays when a payload is being moved. The bay also contains sensors to identify the payload's dimensions as well as safety and motion detectors to identify any passengers left in the vehicle before putting the vehicle into the garage.

2.1.3 Control System

The control system acts as the central coordinator of all subsystems in the garage. In this capacity, it is mainly responsible for issuing instructions to the lifts, bays, and AGVs, maintaining the system's state, and handling requests from the users and responses from the equipment. To ensure robustness, the system must persist the state of the system to enable recovery while also being able to interface with the equipment to query their current locations. Finally, the control system should contain procedures to recover from equipment failures when needed.

2.1.4 User Interface

The user interface provides the means by which the user interacts with the garage. This can be accomplished via a website, mobile app, SMS , or on-site kiosk. In essence, the primary functions are to insert a car, retrieve a car, or schedule a retrieval in the future. These functions are available across all platforms to ensure support to all users regardless of preference or technology aptitude.

2.2 Volley System Design

The components discussed in Section 2.1 are organized according to Figure 2.1. The Volley Garage Manager is the primary software component that directs the overall operation of the garage. The Volley Console application provides an interface for on-site or remote garage operators to make manual changes to the garage (e.g. manually guiding AGVs and payloads, turning off a section of the grid for maintenance, etc.). The garage management software receives information and sends commands to a variety of peripheral components through the fieldbus. Kiosks, cameras, and a number of sensors help to track the current progress of the garage. A master PLC is connected to provide robust control of safety critical features, while a secured wireless networks provides interfaces to the AGVs, lifts, and mobile devices used throughout the garage. Finally, these components all tie into the building's PLC to interface with common components such as fire alarms and security systems.

Figure 2.2 presents the front- and back-end software components developed to support the overall system. At the front-end level, the Volley Design Studio (discussed in more detail in Section 2.6) supports the garage design process. The design studio provides the ability to lay out parking space grids and test them through simulation. Further, multi-objective optimization capabilities are provided to display a multitude of options to the customer with respect to density, cost, and service level.

The Volley Garage Manager (as previously mentioned and discussed in more detail in Section 2.3) is the main software overseer. The manager uses many of the modular blocks from the Volley platform to reliably operate the system. Computationally, the garage manager leverages the job scheduling framework and path planning and routing engines alongside the variety of inventory management systems to reliably insert and retrieve user payloads. To increase resilience, the system is persisted to a local and remote storage layer. As the system matures and is deployed, a machine learning and data analytics layer is added to provide better retrieval predictions and an overall view of the garage's operations. The garage manager then uses a communications interface layer to communicate with the industrial gear that makes up the autonomous garage as shown in Figure 2.1.

The remainder of this chapter discusses Volley's design choices in light of this overall system architecture. Throughout, the problems identified in Section 1.3 are addressed.

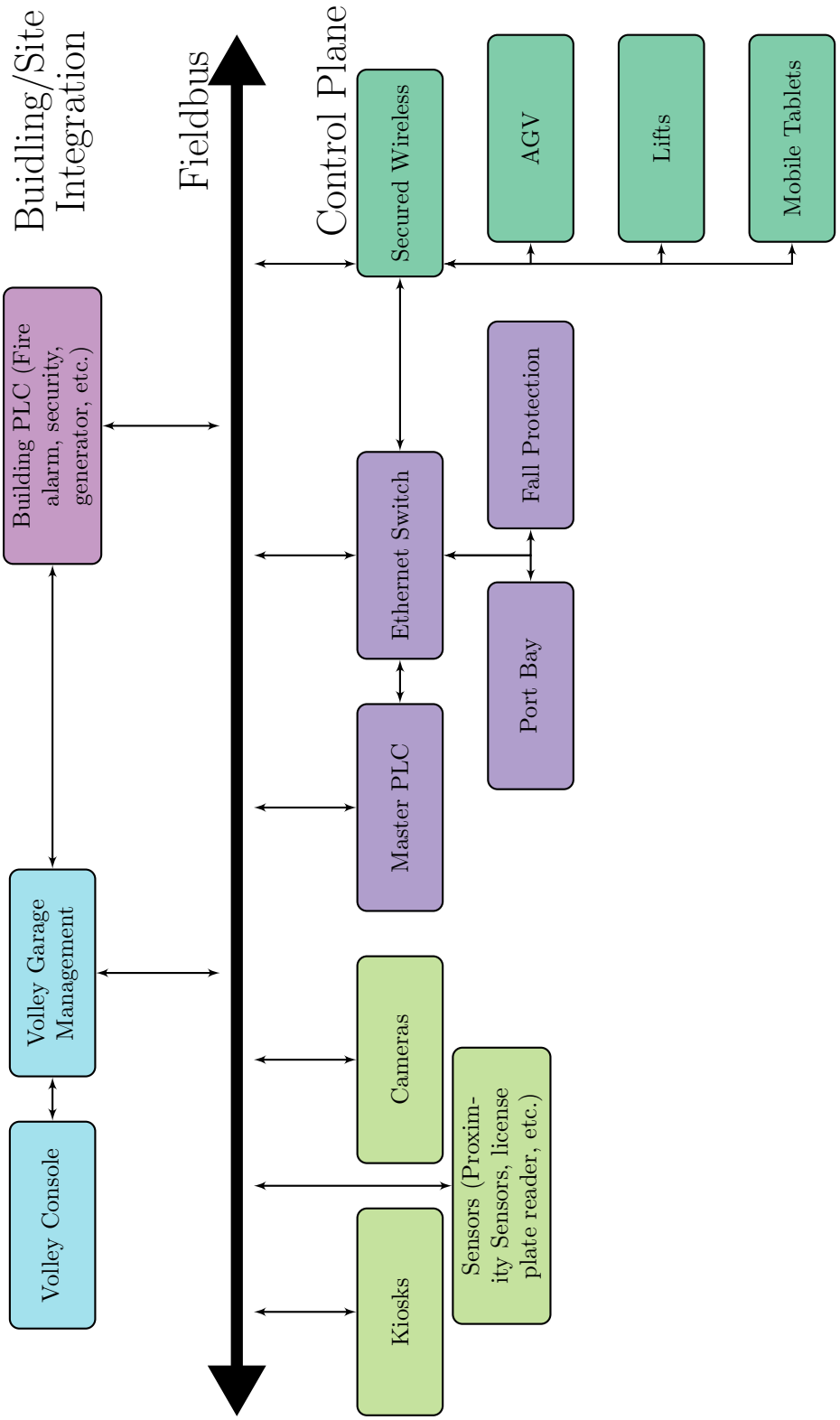


Figure 2.1: Volley System Component Overview

Figure 2.2: Volley Software Architecture

Then, with the design alternatives laid out, Chapter 3 presents an overview of Volley's plan to test against the system metrics of interest.

2.3 Garage Management and Data Analytics

The major gap identified in current systems related to the garage management system (garage manager) is the process by which routes to move the payloads are identified. In most current systems, static routes are pre-calculated and then the system selects one of these routes to follow. Alternatively, A* algorithms have been implemented to increase efficiency but can lead to deadlocks. The remainder of the section details the managers's data structure and logic that is implemented for dynamic, gridlock resistant routing.

2.3.1 Data Structure and Priority Determination

The garage management system stores geometric information as a set of nodes with traversable connecting links. These nodes and links are represented as a mathematical graph that opens a range of simple graph theoretic filters and calculations to determine optimal paths for car retrievals. This structure enables fully dynamic routing of both AGVs and payloads while maintaining conservative policies to prevent gridlocks. Figure 2.3 presents a sample grid and node layout from the Green Street garage.

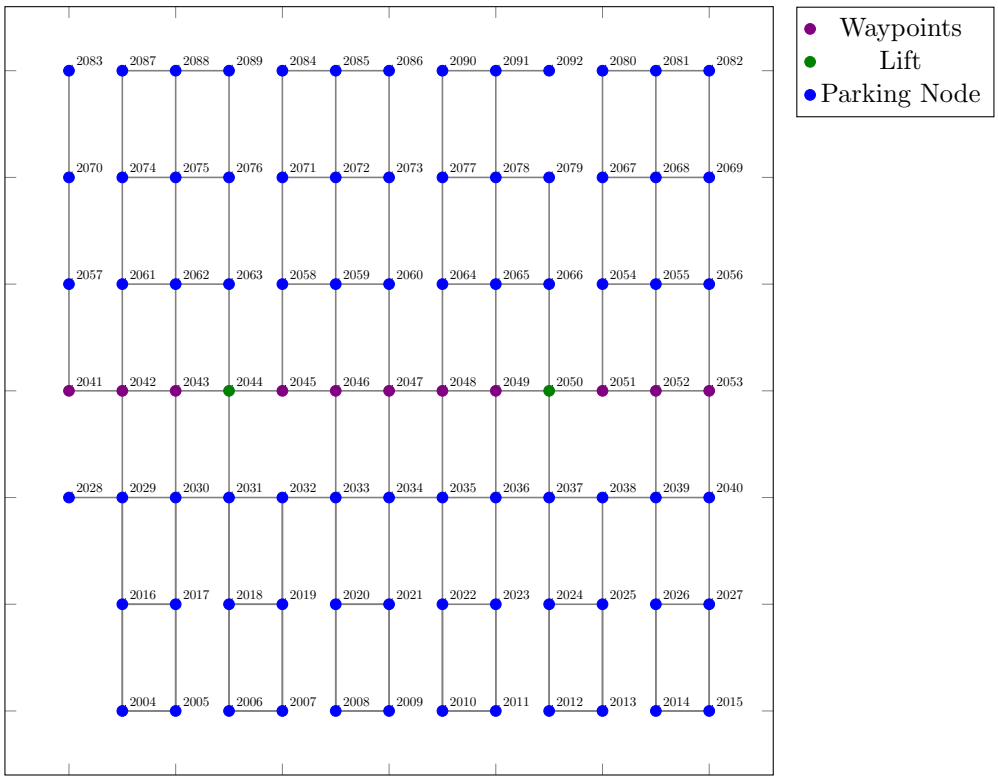


Figure 2.3: Sample Node Layout for Green Street - Level 2

The manager currently recognizes the following types of nodes:

Parking: Parking nodes denote the locations where a car or other payload may be parked. There can be different types of parking nodes that are different sizes to accommodate varied vehicle sizes.

Waypoint: Waypoint nodes are locations in the garage that are intentionally left empty to provide better connectivity from the parking nodes to the lifts and/or bays. In extreme cases, the waypoint nodes may be used as parking nodes, however retrieval times can suffer significantly.

Lift: Lift nodes are the locations on each floor that are serviced by a lift. In essence, lifts can traverse connected lift nodes, and AGVs may only cross a lift node if the lift is present.

Bay: The bay nodes are where customers drop off or retrieve their cars. This is also where the AGV first acquires the car to park it in the garage.

Charging: These nodes are where the AGVs can go to charge.

Lift-Bay: These are special cases of bay nodes that are attached directly to lift nodes (such that that particular bay cannot be serviced by any other lift). This requires special logic to ensure a request from this bay is serviced by this particular lift.

With the graph data structure laid out including the node locations and types with the connections between them, the manager needs to know the order in which to fill the nodes with cars and/or payloads. This static priority determination is necessary to ensure that the garage is utilized most efficiently at any fill level.

Defining some terms is important before proceeding. First, the node's *priority* is simply the order in which that particular node should be filled. Then, a node's *priority tier* represents the number of parked cars that are blocked from the lift when a car is parked in that node given the nodes of a lower tier are filled with cars. Figure 2.4 presents the example layout with the priority tier labeled and colored. The circled node has a priority tier of 2 because with all of the tier 0 and 1 spots already filled, placing a car at the circled parking spot blocks the 2 nodes to its left and right from accessing the lift.

The final definition is a node's current *Pick Configuration* or *Pick Depth*. This details the number of cars that must be moved to retrieve a vehicle. For example, assuming all of the parking spots in Figure 2.3 are filled, retrieving a payload from spot 2059 would only require moving that specific payload. Therefore, it has a pick depth of 1. However, retrieving a car from node 2085 would require moving the cars from nodes 2072 and 2059, so its pick depth is 3. The pick depth is a primary driver of retrieval times, so consistently handling any n-deep pick is a key feature of the management system.

Fill Order Algorithm

The fill order is determined using an iterative filtering algorithm that sequentially identifies the node to fill that causes the least disruption to the previously parked cars. Along the way, additional filters are incorporated to ensure that each node is filled and that early iterations allow for the most nodes to be filled in the lowest priority tier. In other words, the algorithm works to fill the most number of nodes without blocking any other parked cars, then the most nodes that only block a single car, etc. This process to determine the fill priority and priority tier is outlined in Algorithm 2.1.

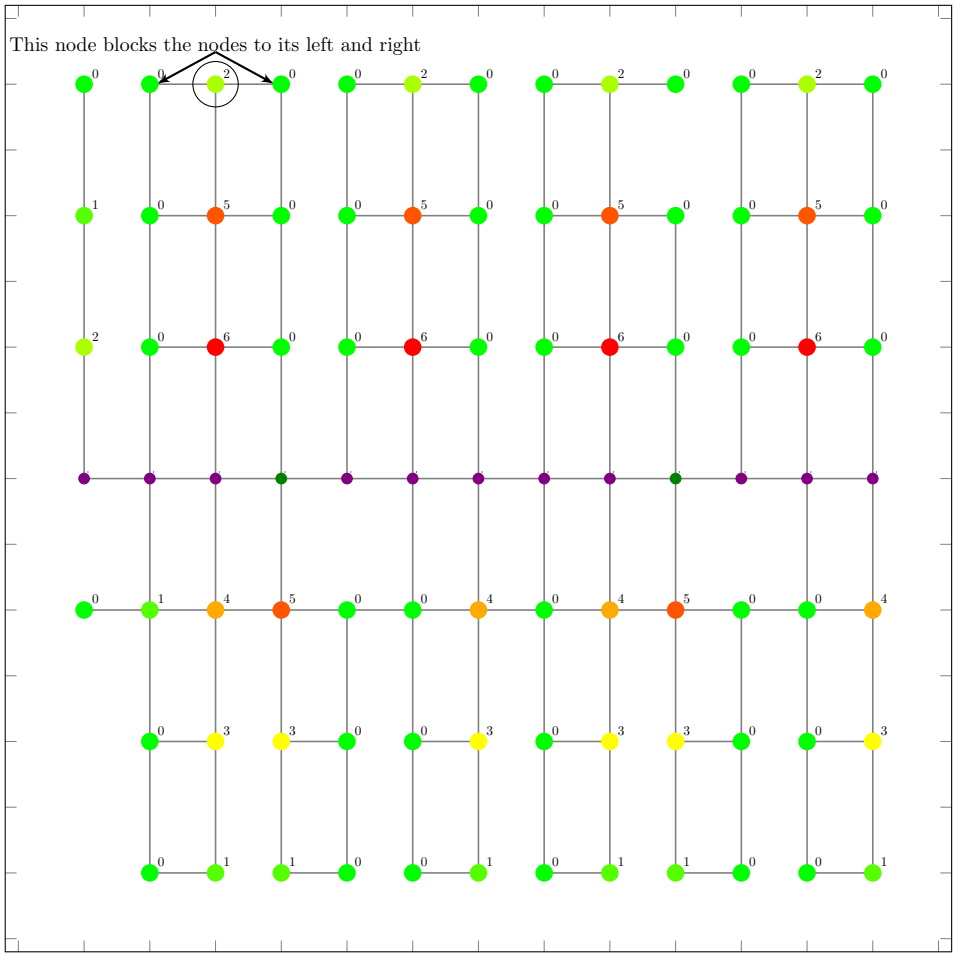
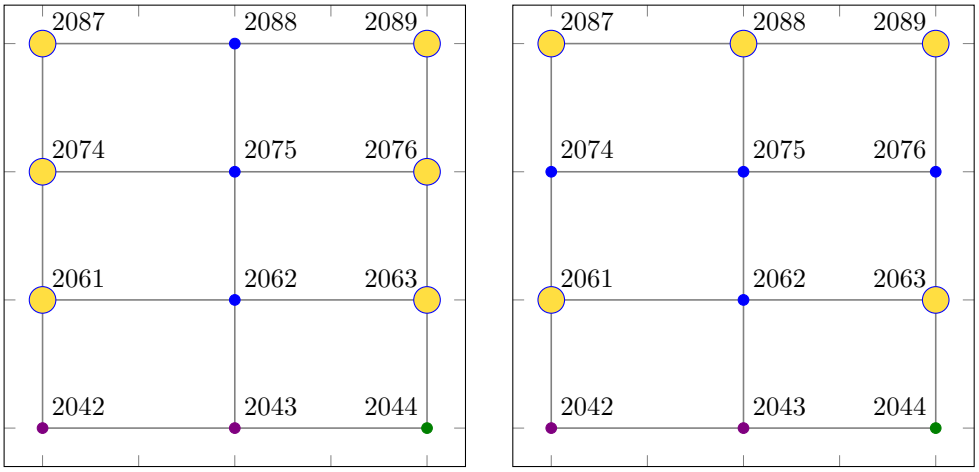


Figure 2.4: Node Layout with Priority Tier for Green Street - Level 2 (Nodes Colored and Labeled by Priority Tier)

The algorithm begins by identifying all of the candidate nodes, which are simply the unoccupied parking nodes. Then, the algorithm calculates the priority tier for each candidate node (e.g. the number of cars that would be blocked if a car was placed on the candidate node) and filters the nodes that are outside of the lowest tier. In other words, only consider the candidates that have the lowest impact to the rest of the parked cars.

The priority tier is the most important of the filters as each node at a given priority tier impacts the retrieval of the same number of cars. Therefore, the remaining filters improve the position of the current insert and work to ensure that the ensuing inserts can be efficient. As such, the remaining candidate nodes are first filtered by their floor/level since lower levels require less time on the lift to access.

The next step is to filter the candidates by the fewest number of adjacent parking spots (filled or unfilled). The idea is that adjacent parking nodes represent spots that could be blocked either now or after a subsequent insertion, so selecting a candidate with the fewest adjacent spots helps to prevent this. As an example, assume the nodes in Figure 2.3 are all free. Based on the priority tier filter, nodes 2087, 2088, and 2089 would all be in the same tier to start (because none of them block any other node from the lift). However, if node 2088 (adjacent to 3 nodes) was filled before 2087 and 2089 (both adjacent to 2 nodes), then the number of spots accessible without being blocked by another car would be decreased. Figure 2.5a shows the optimal nodes to fill in the connected cluster to maximize the number of single pick spaces. Figure 2.5b shows one of the best options for maximizing the number of single pick spaces if node 2088 is filled first. As seen in the example, by filling node 2088 first, the cluster of nodes can only have 5 single pick spaces (Figure 2.5b) instead of the optimal 6 (Figure 2.5a).



(a) Optimal Single Pick Nodes (b) Single Pick Nodes if Node 2088 is Filled First

Figure 2.5: Adjacent Node Example (Selected Nodes Colored Yellow)

After the preference for adjacent nodes is considered, the algorithm selects the nodes furthest from the waypoints to fill first. This is calculated as the number of parking nodes crossed to reach a waypoint that has a path to a lift crossing only other waypoints. The algorithm then also works to fill the nodes that are furthest east and furthest south. Filling the nodes that are the greatest distance from the waypoints, most east, the most south first ensures that all parking nodes are filled. For example, if nodes 2071, 2072, and 2073 are

filled before 2085 in Figure 2.3, then 2085 would not be accessible.

Algorithm 2.1: Node Fill Order Determination

Data: Node Layout and Connections
Result: Fill Order and Priority Rank

for *number of parking spots* **do**
- Candidate Nodes = Remaining empty parking nodes
- Filter by priority ties (Spots that would block the fewest previously filled spots)
- Filter by the lowest floor
- Filter by the spots that have the fewest number of adjacent parking nodes
- Filter by the distance to waypoints that are connected to the lifts
- Select the node that is most eastern then most southern (to provide a consistent fill order)
- Mark the selected node as occupied
- Add the selected node to the fill order list and mark its priority rank based on the number of previously parked cars blocked

end

The algorithm proceeds through this process until all of the nodes are prioritized. The results from the algorithm thus provide the system with first the prioritized list of nodes as well as the priority tier of each node. The tier is used when executing deep retrievals (i.e. retrieves that are more than a single pick) to identify destinations for the blocking payloads. Thus, if forced to move a blocking payload, the new destination should be at the lowest priority tier. The priority level also provides information about the pick types expected at each garage fill level.

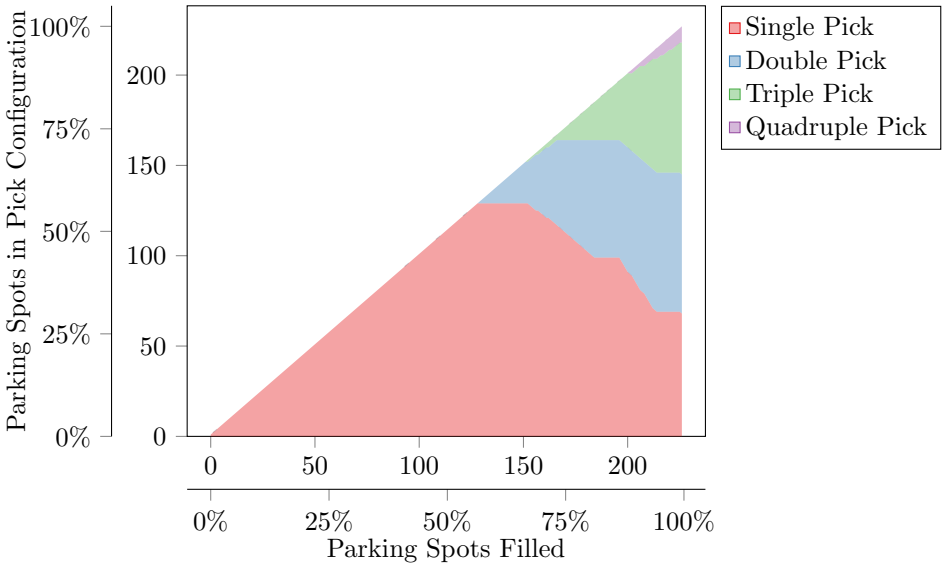


Figure 2.6: Number of Parking Spots in Pick Configuration

Figure 2.6 presents a static analysis of the prioritized layout presented in Figure 2.4. This figure is generated by sequentially filling each of the parking nodes as specified by their

priority and then checking the depth of each filled node. The horizontal axis in Figure 2.6 denotes the number and percentage of parking spots filled. The vertical axis then presents the stacked number and percentage of cars in each pick configuration (Single to Quadruple). This analysis shows that approximately 55% of the parking nodes can be filled before a double pick is required. Then, even when the garage is completely filled, approximately 33% of the nodes are in a single pick configuration and over 60% require only single or double picks. This means that over 60% of the spots can be guaranteed the faster service times available for single and double picks, while the remaining spots may be sold more cheaply for either storage or potentially longer retrieval times. This static analysis is thus a helpful first look to determine the levels of service that could be guaranteed.

The following section describes how the garage manager dynamically categorizes and executes retrievals while avoiding deadlocks.

2.3.2 Payload Retrieval

The most critical feature of the management system is its handling of user retrieval (pick) requests. The time from when the user hits the "Retrieve Car" button to the time the car arrives in the bay is the most important metric for user satisfaction. Previous systems, as discussed in Section 1.3.1, typically rely on static routes in the style of warehouse routing. This section discusses Volley's alternative approach that can handle any deep retrieval while avoiding gridlocks. The section begins by discussing the retrieval types before outlining the strategy to handle each.

Retrievals are classified by the following parameters:

Pick Depth: The pick depth is the number of cars blocking a retrieval.

AGVs Available: The number of AGVs that can be dedicated to a particular retrieval request.

Lift Availability: A retrieval may in some cases have a path to the lift shaft, but that particular lift may be in use by another work item. In this case, cars may be moved to open a route to another lift or the job may wait for the lift to become available.

Open Spots for Blocking Cars: The places where blocked cars may be temporarily placed such that there is a clear path to the selected lift.

The following are the various cases for the open spots to place blocking cars:

- Open parking spots on the same floor without crossing a lift
- Open waypoints on the same floor without crossing a lift
- Open parking spots on other floors
- Open parking spots on the same floor that require crossing a lift to access
- Open waypoints on the same floor that require crossing a lift to access

2.3.3 Retrieval Process

The retrieval process requires first identifying the characteristics of the pick (e.g. depth, AGVs available, spots to move blocking cars, etc.) before determining the best strategy to retrieve the vehicle. Based on the characteristics of the pick, the retrieval type is selected based on a preference identified through simulation studies. The preference for pick type could vary between garages and, in future iterations, may be identified on the fly. With the type selected the code executes the retrieval as discussed in the following sections.

Before proceeding, a few definitions are required. First, a *User Request* represents the complete set of moves required to either insert or retrieve a payload. Essentially, jobs either start or end with a customer interaction. The next planning level is a *Work Item*, which represents the set of moves required to move any payload. Hence, a job may have one work item to move a blocking payload out of the way and another to move the target payload to the customer. Finally, a *Transaction* represents a single set of instructions that are executed by an AGV, lift, and/or bay. Table 2.1 presents the transaction types recognized by the manager. Therefore, the overall system manages the execution of $0 - n$ jobs that contain $1 - m$ work items that have $1 - k$ transactions.

Table 2.1: Transaction Types

Transaction Type	Description
AGV Move	AGV movement without a payload. Can traverse nodes with payloads as long as another AGV is not in that location.
AGV Move with Payload	AGV movement with a payload. Requires that no other payload or AGV occupy nodes on the path.
AGV Lift Payload	AGV is located beneath the target payload and lifts it to prepare for movement.
AGV Drop Payload	AGV lowers the payload it is carrying.
Lift Move	The lift moves to a target floor without an AGV.
Lift Move with Payload	The lift moves to a target floor with a payload but no AGV (the AGV drops the payload on the lift and moves off of the lift before the lift moves).
Lift Move with AGV	The lift moves with an AGV on board. The AGV may or may not be carrying a payload.

Single Pick Retrieval

The single pick retrieval is the simplest retrieval process and forms the base set of moves used for all other retrievals. This represents the simple case where there is a clear and claimable path from the parking location to the pick up bay. The remaining retrieval types essentially move blocking cars or work to reserve AGVs, lift, bays, and/or paths to allow for a "single pick". Once a single pick is available, the algorithm dispatches the AGV that can complete

the retrieval in the shortest amount of time. The sequence of transactions to complete are:

- If AGV is not on the car's floor:
  1. Move lift to AGV's current floor (Lift Move)
  2. Move AGV to lift (AGV Move)
  3. Lift moves AGV to target vehicle's current floor (Lift Move with AGV)
- Once AGV is on the target floor:
  4. Move AGV to target vehicle (AGV Move)
  5. Raise target vehicle (AGV Lift Payload)
  6. Move lift to current floor (if not already present) (Lift Move)
  7. Move AGV and car to the lift (AGV Move with Payload)
  8. Move lift to the bay's floor (Lift Move with Payload)
- With AGV and car on retrieval bay floor:
  9. Move AGV and car to bay (AGV Move with Payload)
  10. Lower car into bay (AGV Drop Payload)
  11. User retrieves car (Job Completed)

To begin the retrieval process, all equipment (AGVs, lifts, and bays) and nodes are locked by the manager for this work item. This naturally prevents gridlocks because no other work item is able to use the equipment or nodes reserved for the current job; therefore, once the job is started, it is guaranteed to have access to the required resources barring any failures along the way (discussed in Section 2.3.4). This strategy is conservative in that the entire path is locked at the start of the job, which could lead to inefficiencies; however, Volley has taken the stance that providing a robust service level through the elimination of potential gridlocks is more important than slight, nominal retrieval time reductions accompanied by significantly more risk. This assumption will be tested in simulation to identify the efficiency losses. To gain back some efficiency, nodes and equipment can be unlocked once they are no longer claimed by the current job (e.g. once the AGV has passed the node and the node is not planned for use by any other transactions in the current job).

With the path and equipment locked, first step is to move the AGV from its current floor to the floor with the target vehicle. Upon moving the lift to the AGV's floor, the AGV begins to move to the lift from its current position. The AGV will only move once the lift is verified to be in place to prevent the possibility of falling down the lift shaft. The AGV also further has sensors to identify that the lift is present before entering the lift.

The AGV is able to move under cars, so when not carrying a payload, it can traverse all nodes other than those that are occupied by other AGVs or reserved by the system for other jobs. Therefore, the move is planned by first filtering the graph to only contain nodes and links that are currently available to the AGV. The management system then seeks to identify the path that requires the least amount of time to traverse. This problem cannot be solved using a simple Dijkstra algorithm [18] as the AGV must accelerate and decelerate along straight line paths while stopping completely to change direction. Therefore, an estimate of the time to traverse a specified path through the network is derived to find the quickest path.

The time to traverse a particular path consists of the time to traverse each individual straight line plus the time to complete each turn. Hence, the time estimation algorithm begins by identifying all of the straight segments in the overall path. Algorithm 2.2 can then be used to calculate the time to traverse a single straight path when provided a path length, AGV max speed, AGV acceleration, and AGV deceleration (potentially different then the acceleration).

Algorithm 2.2: Determination of Time to Traverse Path

Data: Straight path length, AGV max speed, AGV acceleration, AGV deceleration
Result: Time to traverse path

l = path length
$v_{\max}$ = AGV max speed
a = acceleration
a_d = deceleration
t_a = time accelerating to max speed = $v_{\max}/a$
d_a = distance to max speed = $1/2 a t_a^2$
t_d = time decelerating from max speed = $v_{\max}/d$
d_d = distance from max speed = $1/2 a_d t_d^2$
if $d_a + d_d \geq l$ **then** If true, the path length is not long enough to reach max speed. Therefore, determine the time spent accelerating and decelerating:

$$t = \frac{\sqrt{2l}}{a + a^2/a_d} = \text{time spent accelerating}$$

return $t + at/a_d$
else
return $t_a + t_d + \dfrac{l - d_a - d_d}{v_{max}}$
end

The garage management system then uses Algorithm 2.2 to identify the path that requires the least amount of time to traverse. This is done by first finding a set of k shortest (distance-wise) paths using an implementation of the Hoffman-Pavley algorithm, which is a generalization of the Dijkstra algorithm [19]. For the relatively small graph considered, $k = 20$ is sufficient to cover the shortest time path. Thus, the time to traverse each of the 20 shortest distance paths are computed to identify the shortest time path for the AGV to follow.

The AGV thus completes an AGV Move transaction across the shortest time path to enter the lift. The lift then moves to the target floor before the AGV follows the shortest time path from the lift to the target vehicle. The payload is lifted by the AGV and carried to the lift that will transport the AGV and payload to the destination floor (in this case the retrieval bay's floor). An alternative option to investigate would be to drop the payload on the lift to keep the AGV on the current floor and have AGVs on the destination floor complete the remaining transactions. With the payload on the destination level and lifted by an AGV, it is carried to the destination retrieval bay and dropped to be picked up by the customer.

This set of transactions form the base for all moves in the garage. An insert follows the same process but drops the payload into a parking spot instead of the retrieval bay. Similarly, any movement to reconsolidate payloads or move blocking payloads will follow this process. Hence, the remainder of this section discusses how the manager plans moves

for the various retrieval types with the understanding that any individual payload movement follows a variation of this generic process.

Deep Retrieval

When a car to be retrieved is blocked by other payloads, a deep retrieval must be triggered to first clear a path before retrieving the target payload. The preferred method to execute the deep retrieval is to move the blocking payloads to spaces within the garage that are outside of the target payload's retrieval path. Depending on the specific layout of the garage, the space chosen to move the payload to may change between empty parking spots or waypoints that are on the same or different floors.

Regardless of the spot(s) chosen to move the blocking vehicles to, the deep retrieval is executed by first identifying the best path from the target payload to a lift or retrieval bay. The best path is the path with the fewest cars blocking the way, and it is calculated using a Dijkstra algorithm with high weights on links that cross an occupied parking node.

Once the path is selected, the cars to move are identified from the path. Depending on the number of AGVs available and their current locations, one or more AGVs are dispatched to move the blocking payloads to the identified destinations. The destinations are selected such that all blocking vehicles can be moved off of the retrieval path (e.g. it does not make sense to move one blocking payload if the blocking payload behind it cannot be moved off of the path). When the path is clear, the process to complete a single pick for the target vehicle is followed to get the payload to the requesting customer.

Following the retrieval, the blocking payloads may need to be relocated. If they were moved to empty waypoints, then they should be moved to empty parking spaces with the lowest priority tier on the current level or possibly on any level. In the case that the payloads were moved to empty parking spaces, they may just stay there. Even if those spaces are in a higher priority tier than the previous spaces, any subsequent inserts will be placed into the lower tier spaces, so the garage will not be off-optimal for very long. Additionally, a reconsolidation process will be implemented in future versions of the garage management system that can restore the garage to optimality.

If the garage is close to full capacity, there could be times where there are not enough empty spaces to move all blocking payloads. In these instances, it may be necessary to employ a carousel-style retrieval where rows of blocking payloads are shifted in a circular pattern to free the blocked payload. This represents the final fallback retrieval strategy, however initially, the layout is set such that there are always sufficient waypoints to move blocking cars. With more AGVs or faster movement capabilities, this may become a more viable option in future iterations.

2.3.4 Failure Modes and Mitigation

Throughout the operation of the autonomous parking system, hardware and software systems will experience downtime. Properly planning for these events is necessary to ensure consistent and reliable service. Generally, these events result in an AGV, lift, or bay being non-functional. There could also be cases where the entire management system crashed, however this is handled by the software recovering from the database and does not require additional planning considerations.

A hardware failure has 2 consequences on the planning system. First, that hardware system can no longer be seized to complete any work. Secondly, the node or nodes that the down system occupies can no longer be used in any path. For example, if an AGV breaks down, no other AGV can cross its current node until it is moved out of the way.

The planning system is naturally designed to accommodate these 2 possible consequences. First, when selecting the equipment to assign to a particular packet of work, the planning system polls each individual piece of equipment to determine if it can complete the job. In the case of a breakdown, this query returns false for all unavailable pieces, so they will never be assigned work when in a failure state.

Secondly, all routing choices are made on the fly, so when an AGV, lift, or bay go down, their respective node(s) will remain locked. All subsequent routing choices respect this lock and select paths that avoid the locked node. Upon recovery, the piece of equipment informs the manager that it is available for work and, thus, allows the system to move the piece of equipment to unlock the path.

The choices made in the software system provide for a natural means to deal with unexpected equipment downtime. A key to this properly working is that the manager must be reliably informed of the equipment being down. As such, having reliable and fail-safe reporting mechanisms between the hardware and software management system is an important consideration when designing the overall system architecture.

2.3.5 Data Analytics

The garage management system generates a significant amount of data throughout the operation. Because the system has complete control over the movement of all components in the physical system, the data is also clean and well organized. This opens the possibility to apply advanced management and data analytics approaches to the parking structure.

Understanding the wealth of data generated by physical systems represents a key value proposition in today's world. For the autonomous parking garage, there are a variety of stakeholders interested in the available data. Garage operators are interested in the garage's utilization over time and when some pricing or operational change is made. Volley is interested in a variety of data regarding equipment utilization and health, hardware and software performance, and metrics regarding the layout. The remainder of this section discusses these data interests in further detail.

Garage Utilization and Pricing

The owner/operator of the parking garage is interested in understanding the occupancy status of the garage over time. In general, the garage operator is interested in ensuring that they generate the most revenue and maintain customer satisfaction (by minimizing wait times). At a first pass, the operator can examine the entrance and exit data to explore the garage capacity throughout the day and peak times. Comparing this with the user wait times, the operator can determine if changes are required to maintain performance throughout the demand cycle.

Going a step further, the information from the garage can help the owner to properly price parking. At first, this is likely to be a manual process to make changes to the pricing or contract types every couple of months. However, as the system matures, a dynamic pricing model can be adopted to control the garage's utilization while maximizing revenue. In this instance, if the garage is reaching capacity, then the prices could be raised to discourage further customers from entering. Combining this with the data analytics, it may be useful to raise exit prices during peak hours to encourage users to leave before the peak.

System Performance

Volley is very interested in the data from all components of the system to track performance. Upon delivery of a number of systems, there are a large number of components that are producing data every day that can provide insight into potential performance improvements. Much as many of the technology companies deploy many versions of software to run real-time experiments, Volley can choose to modify a routing algorithm or AGV command to test possible improvements. This ensures that Volley continues to provide a high level of service as new cases and situations are encountered.

Remote Monitoring and Predictive Maintenance

Increased density is the primary reason to pursue automated parking; however, additional opportunities to reduce operational costs also present themselves when pursuing an automated valet service. The first way to reduce costs is through remote monitoring of the garage. Current garages require attendants, valets, and/or enforcement agents to ensure a smooth running, profitable venture. These jobs are no longer required when operating an automated system, but the robotic solution does require a human overseer to provide a check on the software and intervene if there is a catastrophic controller failure.

When first implemented, these overseers are stationed at the garage structures to provide possible physical intervention if needed. After the break-in period, the human operators can be stationed in a centralized monitoring location. The monitoring hub has access to all camera feeds from all Volley garages, which allows the operators to assume remote control of any of the systems in the garage. This control center concept is planned for most of the autonomous car concepts, so it represents a good paradigm to manage autonomous systems.

In addition to remote monitoring for safety, all Volley systems will be fitted with sensors to support predictive maintenance. Mechanical systems rarely fail suddenly (even though it may seem like it). Rather, there is typically a gradual degradation of components until one is stressed past its breaking point. Developing systems to sense and procedures to react to these gradual failures is key to increasing up-time, reducing maintenance costs, and improving the life of deployed systems [20].

The particular sensors used to identify component degradation are identified through component-level testing. During this time, a cost analysis is also conducted to ensure that the savings achieved by adding a sensor and processing the data are worth the reduction in maintenance costs. If not worthwhile, component and sub-system level testing can help to identify a periodic maintenance schedule (e.g. time or cycle-based) that controls cost and failure rates.

In the case of an AGV, the primary components that can fail or degrade are the tires, motors, and actuators (basically anything that moves). The wear on these components can be measured either directly or indirectly through wheel encoders, power readings, temperature sensors, and location measurements. During testing, these measurements can be matched against the life and wear on a component (e.g. against direct tire wear measurements). Testing to failure can also identify the range of situations where the components fail. Taking these pieces of information together, models can be generated that estimate the probability of failure given the sensor readings. Then, using the probabilities, a maintenance plan is generated to minimize expected maintenance costs with good up-time.

2.3.6 Section Summary

The garage management software represents a core value proposition of Volley Automation. In leveraging simple graph theoretic calculations with an architecture inherently designed to handle n-deep retrievals and equipment failures, the manager is built to handle challenging situations that could occur in the garage. The following section discusses mechanical design improvements that, when coupled with the improved manager, produce a step change in the autonomous garage's capability and price point.

2.4 AGV Concept Change

The mechanical design of the AGVs in particular can be greatly simplified by decoupling the lifting and support of the vehicle with the vehicle's movement within the garage. The AGV is required to translate the payload in the 4 cardinal directions; when taking inspiration from current warehouse robots, this would be accomplished by first lifting the payload and then moving it on top of the AGV. In doing this with heavy payloads, however, the AGV becomes unreasonably heavy, complex, and expensive.

Volley proposes an AGV system where the trays are fitted with high capacity castors or omni-directional ball transfer units such that the trays bear the load while also being able to translate. Then, the AGV is greatly shrunk and simplified because its only requirement is to release a brake on the tray and tow the tray. As such, the AGV is composed of 4 fixed wheels that can be controlled independently. In this configuration, the AGV can move like a tank by rotating both sides of wheels in the same direction to move forwards or backwards. By counter-rotating the wheels, the AGV can spin in place. In future iterations, the wheels will be replaced by Mecanum wheels, which enable the AGV to drive in all directions without rotating.

The AGV has a large square interface with the tray that provides good straight-line stability once moving. This locks the tray into the orientation of the AGV, releases a mechanical brake on the wheels, and provides an interface for towing. The AGV then tows the tray and payload from the center in its current direction of travel until it reaches the desired location. Upon reaching the location, the AGV lowers the interface to allow turning or movement to another tray.

Volley has constructed a proof-of-concept for this style of AGV (named "Wheelz") that contains less than $3,000 of components. This represents almost a 2 orders of magnitude reduction in AGV cost. At this price point, it is feasible to place an AGV with every parking tray, which could greatly increase the performance of deep retrievals. The implications from this blue-ocean change are discussed in Section 2.4.1. On the other hand, this could bring the garage price point down enough to open significantly more opportunities. Even taking a worst case estimate of $10,000 per AGV, this would reduce the AGV cost for the Brickell House from over $2 million to around $250,000.

The lighter and simpler design also improves the robustness and maintainability of the system. In previous systems, robots or other jacks and rollers are required to move payloads in the system. However, with castors or ball transfers placed on every tray, the trays can be manually moved in a worst case scenario. Furthermore, maintenance on the robots is much simpler because they can be maintained using a depot strategy. With the large size and weight of the previous robots, shipping them anywhere is a challenge; however, the small proposed AGVs can be easily shipped for maintenance by trained, dedicated personnel. Also, with the low cost of the AGVs, it is not a burden to keep spares on hand to replace down AGVs. Then, replacing an AGV is as simple as moving the damaged AGV with a dolly and

placing the spare in its place. In the future, down AGVs could likely be towed out of the way by other AGVs, which greatly reduces the impact of the AGV downtime. In previous systems, if an AGV went down in the middle of an aisle, it would take a significant amount of effort to clear it, which could completely block the retrieval of other cars. In this new paradigm, this process can be completed much more quickly and, even if no other AGV is available, the path can at least be cleared for other retrievals relatively quickly.

2.4.1 AGV Concept Implications - Full Garage Mobility

The "Wheelz" AGV concept provides a blue ocean opportunity for Volley in the autonomous garage space. Autonomous garages leveraging a few high-cost AGVs must contend with a trade-off between purchasing more robots, lowering the density (decreasing the parking depth), and/or accepting a lower customer service level. With the ability to place an AGV under every spot (Full Mobility), many delays in the system can be eliminated. First, a customer's retrieval is never delayed because of AGV starvation; thus, the retrieval bottleneck is moved to free path, lift, or bay availability. The travel time to reach the payload is also eliminated, which could be significant with long distances and/or many turns.

The true benefit of full mobility comes when finding and clearing paths out of the garage. In traditional AGV concepts, clearing a path that is blocked by other payloads is challenging because either the payloads must be all moved one-by-one by a single AGV or multiple AGVs must be deployed. In either case, the retrieval is costly for the system because it either takes much longer or requires a high percentage of the available resources thereby limiting the ability to retrieve other cars.

The full mobility system completely alleviates this problem because all of the blocking spots can be cleared almost simultaneously. Aside from a safety requirement that a blocking AGV must clear the adjacent node before the blocked payload can move, a deep retrieval is now no different than a unblocked pick. This means that garages with high density and very deep retrievals are feasible without compromising customer satisfaction. There are also better opportunities to clear paths to an exit in the full mobility system; thus, retrievals that are considered blocked with traditional AGV systems could possibly be unblocked with complete mobility. Volley has only begun to explore the implications of the full mobility system. Ultimately, this blue-ocean concept provides more capability for a significantly lower cost that will greatly expand Volley's opportunities.

2.5 User Experience

The automated parking system will succeed or fail based on the experience and repeated use of the end customers. The proliferation of web-based service interfaces now enable a firm to better manage and track the experience of the users to support a positive experience. Two factors contributing to customer satisfaction are the price of a service, which helps to manage the service level expectation, and previous interactions with a firm [16]. Given the control over the parking garage, an automated parking system can more directly manage these factors to increase satisfaction across the users.

At any given time in the garage, some cars will be more accessible than others. In the simplest case, a car parked farther away is more difficult to retrieve than one parked closer to the exit. Extending this idea, cars that are blocked by other cars could require significantly more time for retrieval while they wait for a path to clear. Moving cars from the path is also a very expensive operation as this diverts resources from other jobs while potentially blocking other cars. This provides a natural opportunity to discriminate the prices of parking

spots... in the case that the user requires quick access to his or her car, they can be provided with a spot that is guaranteed to not be blocked. However, users who are more flexible can choose to pay a lower amount for a spot that may be blocked and, therefore, may require longer to retrieve. Much as Uber is upfront about the time until pickup, users are happy when they receive the level of service that they pay for [17], so a tiered pricing system can work to improve satisfaction while still supporting high parking density.

While the goal is to always meet the service level provided, there will be instances where the system cannot meet expectations. In this case, the garage operator must work to repair the relationship with the consumer to support a repeat visit. In an experiment conducted in conjunction with Uber, Halperin et al. explored how apologies influence future use of the service [21]. Their findings indicated that monetary apologies helped to retain customers, and that after repeated apologies, customer retention began to drop. In essence, users can be retained through monetary incentives but do eventually lose trust in the firm. The automated parking system can work to ensure that a customer with previously poor experiences has a positive experience by prioritizing his or her retrieval and, to help secure repeated use, can offer free parking on the next trip following a poor experience.

Allowing the user to schedule a pickup can also help to improve satisfaction. If the users are reliable, the actual retrieval time would become secondary to the system's ability to provide the car at the requested time. Through properly sizing a buffer near the exit, the system can queue cars for user requests ahead of time to provide instantaneous service once they arrive at the garage. Again taking from Uber, the system may add a penalty if the customer is late either as an extra charge or revocation of the pre-scheduling privilege in the future.

By considering the user in the system's design, Volley seeks to better utilize the engineered solution. Discriminating users into service tiers allows the system to operate at its designed efficiency without needing to dedicate an exorbitant amount of resources to deep retrievals. Ultimately, this consideration provides another dimension that can help the developer to increase density while maintaining satisfaction without the increased cost of more bays, lifts, AGVs, personnel, and parking spaces.

2.6 Design Process

Four shortcomings of former companies' design processes are identified in Section 1.3.4: 1) Inadequate system robustness studies conducted through simulation, 2) Simulation does not interface with the garage controller, 3) Traditional proposal and project design process ignores additional considerations brought by autonomy, 4) Lack of thorough system integration testing. The remainder of this section presents Volley's approach to resolve these deficiencies.

2.6.1 Robust Design Methodology

A positive user experience, as discussed throughout this work, is based solely on quick and reliable payload retrievals. From a simulation perspective, an autonomous parking garage is relatively simple and well-understood; thus, simulation can provide a helpful digital testbed to examine how the controller is able to deal with a wide range of scenarios and situations. Previous queuing and simulation studies mainly focused on the nominal case (e.g. all AGVs are operational and the demand is not excessive), which does nothing to examine the robustness of the system to equipment outages or excessive demand. Therefore, Volley proposes to leverage concepts from *robust design methodology* during system development and testing.

The goal of robust design is to develop a product whose performance is insensitive to noise and variations [22]. The "noise" in the automated parking system encompasses changes in demand, equipment downtime, or node downtime, while the "variations" are the choices that can be controlled (e.g. number of lifts, node layout, etc.). Designing robustness into the system thus requires testing the system against a variety of cases in different parking structure layouts. As opposed to the queuing and nominal load simulation studies carried out by previous companies, Volley is expanding the cases covered by the simulation to those presented in Table 2.2.

Table 2.2: Simulation Scenario Sets

Scenario Set	Description
Anticipated Requirement	This is the baseline scenario with the design load (90^{th} percentile day) to test the nominal performance. A case where the load is multiplied to 125% is also investigated.
Test Data Set	These scenarios are comprised of a set of test insertions and retrievals created either from gathered historical data or designed to stress a particular aspect of the system (e.g. how does the system handle 10 simultaneous retrievals).
Equipment Downtime Sensitivity	This scenario investigates the system performance's sensitivity to the mean time between failure (MTBF) and mean time to repair (MTTR) of the equipment in the garage.
Equipment Availability Sensitivity	This scenario set tests the performance of the system with different numbers of lifts, bays, and AGVs.
Equipment Speed and Acceleration Sensitivity	These scenarios explore how the speed and acceleration of the AGVs and lifts impact the system performance.
Evacuation Time	In this scenario, the garage is first completely filled, and then each retrieval is triggered simultaneously. Essentially, this is a worst-case scenario for the controller and tests the total time it would take to empty the garage.
Fill Time	This scenario tests the total time it would take to fill the garage. While not as interesting as the Evacuation Time case, this again tests the limits of the controller.

The scenario sets presented in Table 2.2 cover the range of noise and variations in the automated parking garage system. The examination of a higher than anticipated demand along with the fill and evacuation scenarios provide insight into how the system handles excessive demand. The sensitivity to equipment downtimes investigates robustness to unexpected breakdowns, while the availability and speed/acceleration studies examine how the performance is impacted when the number and capability of each piece of equipment are

varied. Overall, these scenarios, when evaluated across a wide number of layouts, provide a holistic picture of the performance and robustness of the designed controller and system in general. Once validated against real-world performance, these simulation studies and associated, automatically-generated simulation reports provide an important tool to evaluating the viability of new projects.

2.6.2 Simulation-Controller Interface

The next improvement to the design process works to leverage the simulation framework as a digital testbed for the system controller. This represents an extension of the typical use of simulation during the project proposal process. Generally, the simulation study is limited to essentially testing whether the proposed garage layout can support the expected traffic. This is accomplished by mocking the controlling logic in the simulation as opposed to actually integrating the true controller with the simulation framework. This misses an opportunity to expand the claims resulting from the simulation study from "a generic control strategy can support the requirements" to "the specific controller software and interface supports the requirements."

Volley thus is leveraging the core scheduling and dispatching software to control the equipment in the simulation. The commands are sent to the simulation using the same interfaces that are used to control the real-world components. As such, the simulation acts as a hardware emulator, which should work to reduce future integration challenges. Figure 2.7 illustrates the interface between the simulation and the controller for a single AGV.

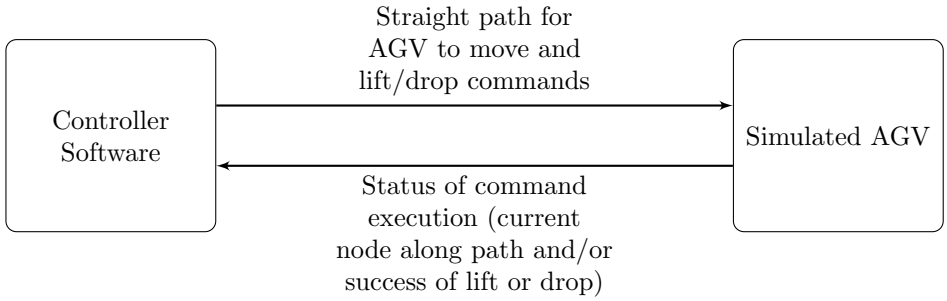


Figure 2.7: Controller–Simulation Interface for an AGV

By employing the actual interfaces to control the simulation, many more aspects of the system can be tested and optimized before deployment. For example, logical bugs in the controller can be identified and corrected earlier and more reliably. Different control strategies and updates can also be more easily tested and, once proven in simulation, can be rolled out to the real world with more confidence. The simulation framework may also help during the execution of the real system by enabling the controller to "look ahead" to the potential future and make choices that lead to more desirable outcomes. For instance, given the current state of a garage, a set of processing work items, and possible future requests, the simulation could be used to potentially re-prioritize the work items or identify a good time to reconsolidate the garage. In the end, the tight integration between the simulation and the controller enables greatly expanded digital testing than traditional, one-off simulations.

2.6.3 Project Design and Layout

The design process for this system is challenging because while every instance of the automated garage system has similar components, there are differing layouts and user requirements for each. This means that the electromechanical and control systems must be designed to cover the range of expected implementations, while the specific layouts and numbers of AGVs, lift, and bays must be optimized for each individual project. The design and testing of the controller and electromechanical systems should cover the range of projects (and are tested against the variety of layouts as discussed in the previous sections); however, the specific layout must be selected through close interactions with the developer.

A project begins with the developer providing Volley with a structural outline for the parking structure. With this as the starting point, Volley proceeds through an automated design process to identify the best options to meet the parking requirement, user expectations, and cost requirements. Design automation and extensive simulation-based testing are built into the core of the design process to quickly provide customers with defensible options for their implementation. This speed and thoroughness represents a differentiating capability that enables Volley to identify the best design options within a fraction of the time of competitors.

This remainder of this section discusses the utility of a parking layout and the general process by which Volley works with the developer-customer to design a parking area and lay out the grid of parking spaces, lifts, and bays. Along the way, the AutoCAD design plug-in is discussed that helps the designer to place parking spaces in a provided parking area. Then, the selection of node types and simulation studies are discussed. Finally, the capability to optimize the node types and provide quantitative decision-support to the developer is presented. In all, this process enables the developer to work with Volley to identify the options and trade-offs available to make the best selections for their budget and user requirements.

Design Utility

The utility of an autonomous parking structure is composed of the structure's size/parking density, customer experience (e.g. wait time), garage throughput, and cost as discussed in Section 2.1. These utility metrics are inherently at odds: fewer parking spots and more AGVs will reduce the customer wait time but increase costs and reduce density. As such, instead of a single optimal point, there is a *Pareto frontier* of non-dominated solutions that must be examined, filtered, and weighted by Volley and the customer [23]1.

Figure 2.8 is provided to explain the concept of Pareto optimality. In the figure, Y_1 and Y_2 are objectives to minimize and the points represent potential solutions. The blue points along the curve are the non-dominated points because there are no other solutions that are better or equal in both dimensions (e.g. improving in one dimension degrades the solution in another). The red points, however, are dominated by points along the Pareto-efficient frontier because there are other solutions that are better in both dimensions.

The concept of Pareto optimality can be expanded to n dimensions such that a Pareto efficient solution is one that is not dominated in all dimensions. As such, the product of the design process are a set of Pareto optimal solutions that can be sorted and filtered by the customer to identify their preferred solution.

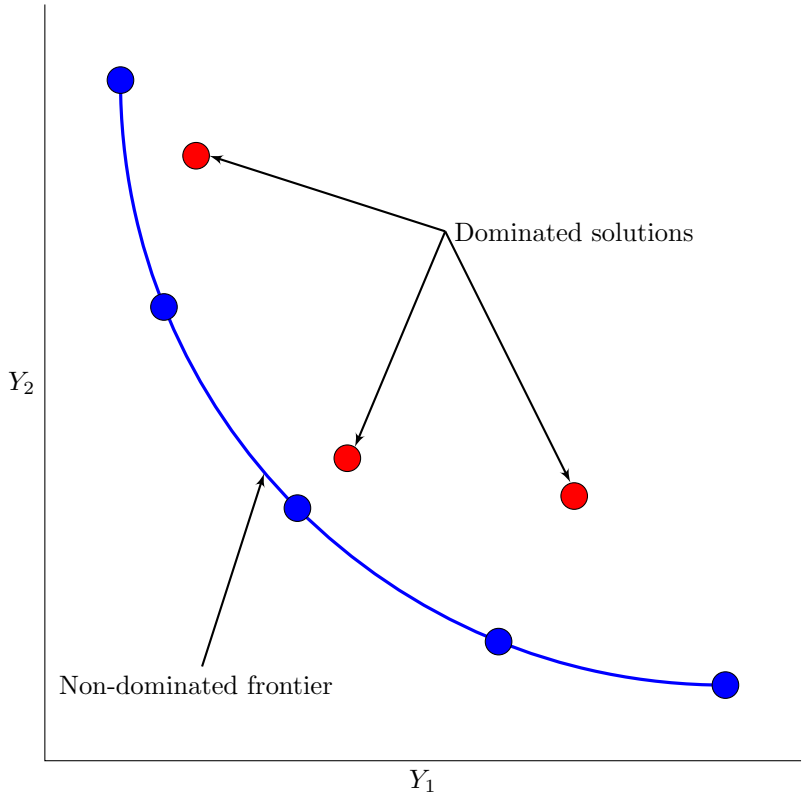


Figure 2.8: Pareto Optimality Example

Design Process

The overall design process is presented in Figure 2.9. The process begins with a blank piece of land for development or a building shell for integrated parking. This is provided to Volley as an AutoCAD/Revit drawing (or Volley can create the drawing in house). At this point, the area for parking is defined along with building obstructions (e.g. columns, stairs, etc.). From here, an AutoCAD/Revit plugin is used to automatically fill the parking area with parking spots to maximize the number of potential spaces (Example in Figure 2.10). This plugin further supports manual tweaks to the layout as there are likely considerations or options that are not properly considered by the automated algorithms.

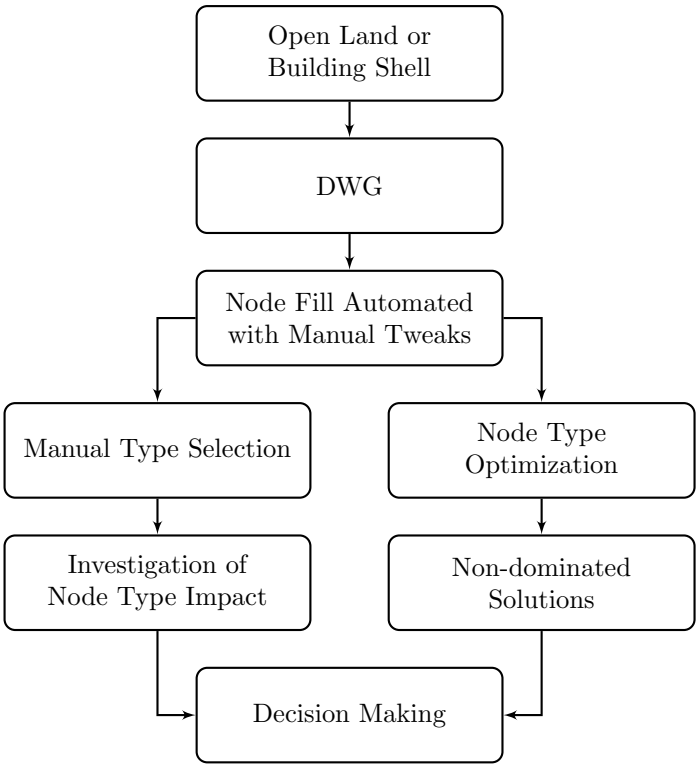


Figure 2.9: Overview of the Parking Layout Design Process

With the node locations defined, the next step is to select the type of each node (e.g. parking, lift, bay, waypoint). In the first pass, each node type is the same size, so the node types can be parametrically changed and run through the simulation to estimate the impact of the change. For instance, a set of parking nodes may be changed to waypoints to increase connectivity in the graph → improving retrieval time but reducing the number of parking spots. A single simulation run requires about 5 minutes to complete, so near real-time feedback between the node type selection and its impact on retrieval time is available. This supports the customer by allowing him or her to gain a qualitative understanding of how changes to the node types can impact system performance. With a base examination of the node types, a multi-objective optimization routine is leveraged to identify the Pareto-efficient solutions for the particular node layout.

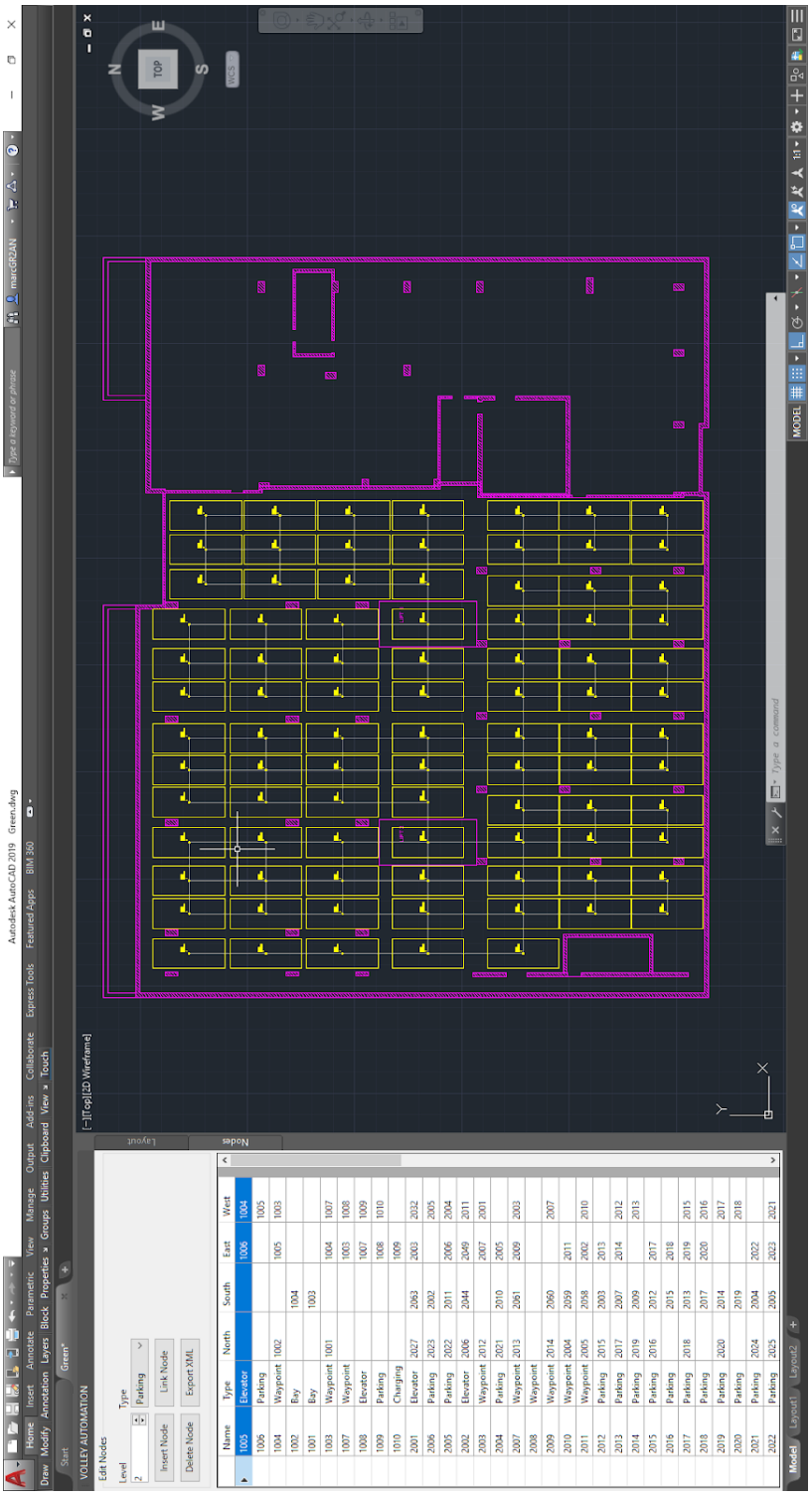


Figure 2.10: Layout Created in AutoCAD Design Studio Plugin

Multi-objective Optimization

The individual design problem can be formulated as a multi-objective optimization routine. The optimization problem is formulated as:

$$
\begin{array}{lcl}
\max & \text{Parking Spaces} & \sum_{i=1}^{N} (x_i == \text{Parking}) \\
\min & \text{Parking Structure Size} & \max(\text{Levels}) \\
\min & \text{Customer Retrieval Time} & \text{Evaluated from Simulation} \\
\max & \text{Garage Throughput} & \text{Evaluated from Simulation} \\
\min\limits_{x_{\text{AGV}}, x_i} & \text{Cost (AGVs, Lifts, Bays)} & \sum_{i=1}^{N} (x_i == \text{Lift}) * c_{\text{Lift}} \\
 & & + \sum_{i=1}^{N} (x_i == \text{Bay}) * c_{\text{Bay}} \\
 & & + \sum_{i=1}^{N} (x_i == \text{AGV}) * c_{\text{AGV}}
\end{array}
\tag{2.1}
$$

$$
\begin{aligned}
\text{S.T.} \quad & 1 \leq x_{\text{AGV}} \leq \text{Max AGVs} \\
& 1 \leq \sum_{i=1}^{N} (x_i == \text{Lift}) \leq \text{Max Lifts} \\
& 1 \leq \sum_{i=1}^{N} (x_i == \text{Bay}) \leq \text{Max Bays} \\
& x_i = \{\text{Lift}, \text{Bay}, \text{Waypoint}, \text{Parking}\} \\
& N = \text{Number of Nodes}
\end{aligned}
$$

In this problem, there are 5 objectives under consideration: 1) maximize the number of parking spaces, 2) minimize the parking structure size (usually manifested as minimizing the number of levels), 3) minimize customer retrieval time, 4) maximize the garage's throughput, and 5) minimize cost (minimize the number of AGVs, lifts, and bays). The problem is constrained to have at least 1 bay, AGV, and lift (assuming multiple levels are required) while remaining below a maximum value for each. Further, each x_i must be set as a lift, bay, waypoint, or parking node. Considerations for the lift-bays can be added as needed. Finally, N is equal to the total number of nodes.

There are 3 basic methods to solve multi-objective optimization problems. These approaches can be classified as *a priori*, *a posteriori*, or interactive. The *a priori* approach works to reformulate the multi-objective problem into a single objective [24, 25]. This can be accomplished in multiple ways, such as combining the objectives into a single weighted-sum function, transforming all but one objective into constraints, leveraging techniques such as Goal Programming, etc. [24, 26]. The *a posteriori* approach, conversely, identifies a family of Pareto optimal (or non-dominated solutions). Following the optimization, the decision-maker(s) sort through the results to select a compromised solution to implement. Finally, the interactive approach brings the decision-maker "in the loop" to update preferences and constraints as the optimization is progressing [24–26].

Of the proposed multi-objective approaches, the *a posteriori*, Pareto optimal-based solution approach has many benefits over the others. Unless the user has a large amount of experience with the problem, the *a priori* approaches can produce solutions that are

not preferable. Defining weightings among the objectives, constraint values, and/or goals to produce desired results is difficult [24–26]. Furthermore, interactively updating preferences is difficult when objective evaluations are time-consuming [26]. It is also challenging to ensure that the changes to the preferences will produce the desired results [25]. The *a posteriori* method avoids these challenges by providing the decision-maker with a quickly sortable population of non-dominated solutions. In this way, the decision-makers have access to all available trade-offs and can quickly understand how their preferences impact the final selection [24].

However, like any approach, the *a posteriori* approach presents some drawbacks. First, identifying an estimate of the entire Pareto-optimal set requires significantly more function evaluations than either the *a priori* or interactive approaches [24, 26]. This can be acceptable for longer-term problems that do not need to be solved immediately. The additional time can also be reduced through efficient modeling and parallelization [27, 28]. Second, when faced with hundreds to thousands of non-dominated solutions, the planners can become overwhelmed with the shear number of possible solutions [24, 26]. Hence, it is recognized that efficiently generating the Pareto frontier and effectively visualizing, analyzing, and down-selecting the non-dominated points are the main challenges in the *a posteriori* paradigm [24, 28, 29].

Volley thus proposes the use of a meta-heuristic, *a posteriori* multi-objective optimization algorithm to efficiently identify points along the Pareto frontier. *A posteriori* multi-objective describes the method by which solutions are compared, but this does not discuss the strategy taken to identify the next solutions to try. At a high level, there are 3 classes of strategies to guide the optimization algorithm: 1) Exact, 2) Heuristic, and 3) Meta-heuristic methods [26, 28, 30–33]. Exact optimization techniques, such as linear programming, branch and bound, or sequential quadratic programming, are able to mathematically prove that an optimal solution is identified. However, these techniques require an exorbitant amount of computation time for complex problems and require a well-defined formulaic relationship between the inputs and outputs. Hence, these techniques are not applicable to the present case that relies on simulation to estimate the performance of a solution.

Expanding beyond exact methods, heuristic methods attempt to apply rules-of-thumb to iteratively move towards an optimal solution. An example is a local search where the inputs are varied individually and then the modified solution is selected if it is better than the current. This approach supports more complex problems but does tend to get stuck in local optimums. To remedy this, the meta-heuristic approaches wrap lower level heuristics into an overarching global search strategy. For instance, the algorithm may accept movement to a worse solution to encourage exploration of the optimization space. Examples of these algorithms include simulated annealing, tabu search, and genetic algorithms.

Multi-objective Genetic Algorithm

The particular problem lends itself well to a genetic algorithm-based search because 1) the solution space is complex with many local minima and 2) the evaluation requires a good deal of time but is parallelizable. Genetic algorithms mimic Darwinian evolution and work to evolve a population of solutions towards an optimum. Figure 2.11 presents a generic overview of genetic algorithms. The algorithm starts by initializing a population of potential solutions. The inputs for the initial population are usually set randomly. With this population created, the *fitness* (objective function value) of each individual is calculated. If the stopping criteria has not been met, then parents are selected from the population for crossover. The crossover process mimics the chromosome blending that occurs during

biological reproduction by combining portions of the parents' decision variable vectors into 2 new children. These new children typically then replace their parents in the population. A mutation step then occurs where parts of some population members' decision variable vectors are modified randomly. This helps to ensure genetic diversity while also providing small changes that can hone in on optimal solutions. Once the crossover and mutation steps are completed, the updated population is evaluated and the process continues until reaching the stopping criteria.

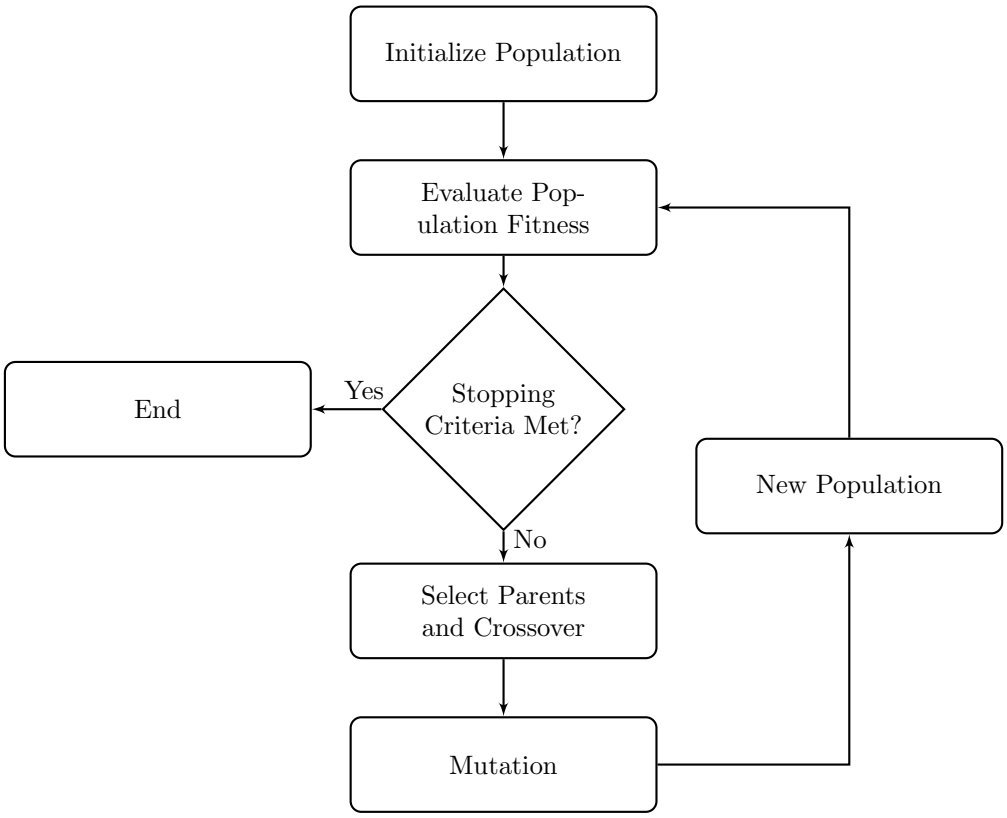


Figure 2.11: Genetic Algorithm Process

The basic genetic algorithm framework is extremely flexible to support a number of objective function evaluations, heuristic crossover and mutation strategies, and selection strategies. Adapting the algorithm to support the multi-objective search of interest involves replacing the typical fitness evaluation (e.g. single performance number) with a ranking based on how close a particular solution is to the current non-dominated set. The strategy chosen for this is called the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [34]. This algorithm is a standard genetic algorithm variant for multi-objective problems.

The NSGA-II ranks solutions based on first their Pareto rank and then by a crowding distance metric. The solutions with the best Pareto rank are those that are in the globally non-dominated set. The next rank is composed of those solutions that are only dominated by the solutions in the first Pareto rank. This continues until all solutions are ranked. Within each rank, solutions are ordered by their crowding distance, which is the distance of each solution to their nearest neighbor in the same rank. Essentially, this prefers solutions

that are not close to other solutions to better encourage exploration of the Pareto frontier. These 2 values are used to select the individuals to mate and/or be mutated for the next generation.

The next generation is typically selected via a tournament method whereby a small number of "combatants" are selected for a tournament. The combatants then complete pair-wise battles where the winner is selected based on the combatants' fitness values (i.e. the combatant with the better fitness wins). For the NSGA-II, the winner either has a better Pareto rank or, if the combatants are in the same rank, the winner has a better crowding distance. The winner of the tournament is then added to the reproduction pool, and tournaments are run until the pool for reproduction is filled. With the parents selected, the algorithm proceeds through crossover and mutation before evaluating the new generation. This continues until a stopping criteria is met, which is commonly simply a maximum generation count.

With a set of non-dominated solutions identified via the NSGA-II, the following section discusses the process and tools used to collaboratively down-select the layout to implement.

Multi-Criteria Decision Making and Data Visualization

Many problems faced in today's world require a multi-disciplinary team to arrive at a solution that satisfies all sides of the issue [35–38]. However, in situations where the users may have different backgrounds and preferences, each user may prefer or be more comfortable with one view over another [38]. Such situations contribute to making collaborative decision-making and interactive data visualization techniques key enablers to the identification of solutions to complex, multi-faceted problems [38–43]. Indeed, when multiple views or representations are required, ensuring that each are interconnected is necessary to support collaboration across disciplines [38]. Therefore, identifying views and decision-making strategies familiar to each user can help to facilitate down-selection. In support of this need and following the *a posteriori* method [24], general multi-attribute decision-making principles [29, 44, 45], and techniques supporting the visual analytics mantra (Analyze first – Show the important – Zoom, filter, and analyze further – Details on demand)[46], this section reviews and discusses techniques for the system-level visualization and analysis of multi-dimensional data to enable: 1) system-level understanding of the trades and analyses available, 2) scenario selection and comparison, 3) and detailed investigation and modification of a small number of solutions.

System-Level Visualization and Analysis

Identifying a proper way to visualize, filter, and down-select between options is necessary to enable humans to digest and understand large, multi-dimensional design and decision spaces [29, 47, 48]. An interactive scatterplot matrix is identified by Stump et al. as a strong visualization tool for this type of analysis [29]. The relevant strengths of scatterplot matrices (reproduced from [29]) include:

- Ideal for design spaces with few dimensions but many data items
- Useful for the visualization of correlations and trade-offs
- Used as a region query by selecting interesting data points

The problem of interest matches these characteristics as there are a limited number of response dimensions with strong correlations to examine. In such instances, the capability of scatterplot matrices to provide visualizations of pair-wise correlations, trade-offs, and regions

of interest can significantly increase the understanding of the human decision-maker [29, 49]. However, a scatterplot matrix may still present an overwhelming amount of data that provides little insight into which point(s) or region(s) of points represent "good" solutions. Multi-attribute ranking algorithms can be integrated to alleviate this issue and enable the dynamic ranking and coloring of Pareto-optimal solutions based on user-defined preferences [29]. In particular, integration of visualization and multi attribute/criteria decision making capabilities would allow users to visualize the impact of their preferences on the decision space and identify well-performing points for more detailed scenario comparison [29].

A popular multi-attribute/criteria decision-making (MADM/MCDM) method that fits these requirements is the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [50, 51]. The underlying principle driving the TOPSIS algorithm is that the "best" solution should be geometrically closest to the ideal solution (e.g. the individually best values for each objective function) and farthest from the negative ideal solution [52]. An example of this is shown in Figure 2.12. The green point in the lower-left corner represents the ideal solution while the red point in the upper-right corner is the negative ideal solution.

The TOPSIS algorithm begins by normalizing and weighting each response. The weighting allows the user to dynamically modify their preferences (e.g. prefer lower costs to more parking spaces) and immediately see how the recommended responses change. With the weightings set, the algorithm essentially scales the axes such that the responses of interest are closer to the ideal solution. For example, in Figure 2.12, if Y_2 is preferred, this axis is expanded such that variations in Y_2 are more pronounced than variations in Y_1. After this transformation, the Pareto point with the minimum Y_2 value is closer to the ideal point than that with the minimum Y_1 value.

The scatterplot matrix and TOPSIS ranking algorithm can thus be used in concert to identify solutions of interest. This ranking analysis is conducted in real-time with Volley and the customer in the room. In doing so, the impact of preferences and requirements can be immediately quantified to support more informed decision making. As promising results are identified, more detailed metrics about the selected solution are presented to the user, and a full simulation suite can be executed to examine all facets of a solution.

2.6.4 System Integration Testing

The final shortcoming to bridge in the design and integration process is related to system testing prior to deployment. In general, the autonomous garage can be rigorously simulated then tested to ensure consistent, reliable performance upon deployment. Unfortunately, previous attempts have failed to conduct adequate system testing and discovered problems during deployment. Given the stain that currently plagues autonomous garage systems, ensuring the product can reliably provide the expected service level is necessary.

Volley's systems design and testing process is modeled after the System's Engineering V-Diagram [53, 54]. During design, the system is continually broken down into smaller components that each have their own requirements. For example, the overall system shall be capable of inserting a payload into the garage. To accomplish this, the AGVs must be capable of moving the payloads. To move the payloads, the AGV motors must have a certain horsepower and so on.

Figure 2.13 presents a version of the systems engineering V diagram. The main purpose of the V Diagram is to ensure that testing is completed against requirements at each level of the design. Once the system is designed and decomposed down the left side of the V, the hardware and software components are integrated and tested. For the present system, this could involve testing that the code on the AGV properly responds to directions. Then,

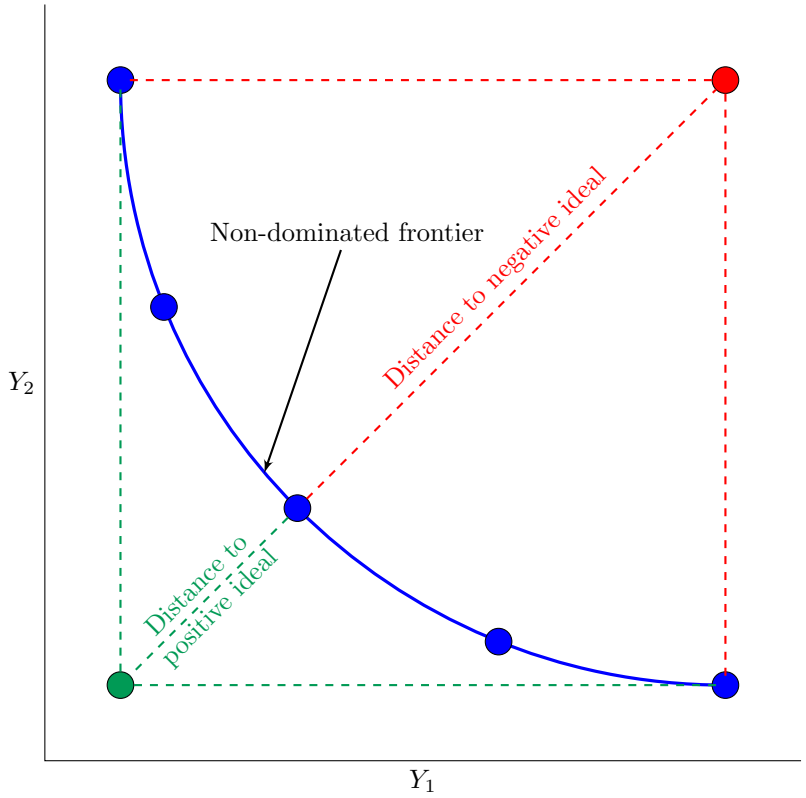


Figure 2.12: TOPSIS Example

moving up the right side of the V, the interface between the AGV and controller could be tested to ensure that the AGV is properly controlled by the controller. Moving further, the ability of the controller to supervise all equipment in the garage is tested. This would complete the verification process (i.e. the system is performing all of the functions that it is designed to perform).

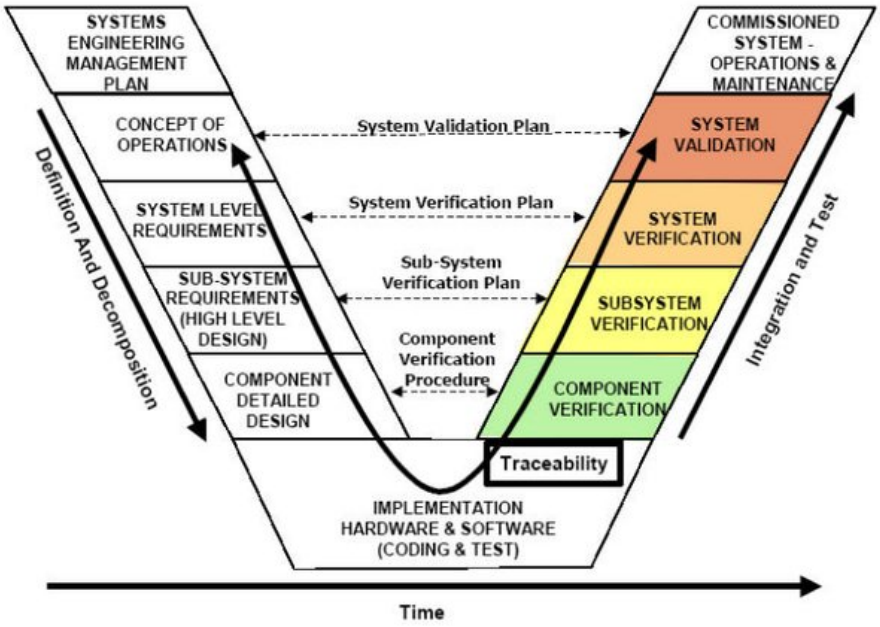


Figure 2.13: Systems Engineering V Diagram (Reproduced from [54])

The final step is then the validation process. Contrary to verification, validation tests to ensure that the system accomplishes its goal as outlined in the concept of operations. The purpose of an autonomous parking garage is to insert and retrieve customer vehicles in a specific amount of time. Even if the system operates as designed (i.e. is verified), this does not necessarily mean that it can fulfill the concept of operations (validation). These validations tests require fully integrated systems to concretely explore the system's performance.

As such, a key principle in Volley's design and integration process is to ensure plenty of time for full-scale integration testing before opening to customers. One benefit of the current system is that it is easy to test at full scale. Inserts and retrieval requests can be sent from a testing system, and the AGVs, lifts, and bays can manipulate empty trays assuming that vehicles are present. As such, true retrieval times can be gathered for a range of demand scenarios. This also can provide valuable information on the reliability of the system and sub-systems. Therefore, Volley intends to push for ample integration testing time and resources for all projects. As the system matures, the resources and time required will decrease, but especially with previous failures and the customers' high expectations, this cannot be neglected during the proposal and planning phase. Chapter 3 covers Volley's integration and testing plan in further detail.

2.7 Additional Technology Developments

Throughout the development of Volley's initial technology base, additional improvements are identified that help to expand the utility of the solution. While not directly targeting identified deficiencies in the industry, the following section discusses these development efforts that are geared towards improving the customer experience, system safety, and operational cost.

2.7.1 Retrieval Request Prediction

During normal garage operations (e.g. no breakdowns), a customer's retrieval is negatively impacted when resources are in use for other jobs or the customer's car is blocked by other cars in the garage. If retrieval requests can be predicted ahead of time, then these factors could be mitigated by staging cars for retrieval either by unblocking them or moving cars expected to be retrieved to a holding area close to the exit.

Accurately predicting the time of a retrieval can be accomplished in a variety of ways. As discussed previously, if the retrieval is scheduled far enough ahead of time, then the scheduling information could be used to identify the cars to stage. Another similar option is to have a pre-paid system where the customer pays for a particular amount of time. Then, customers who have less time left could be staged for retrieval.

Barring a method to reliably ask all customers to schedule or pre-pay for their parking, statistical analysis and machine learning techniques can be employed to predict requests. The category of the parking garage (e.g. condo, hotel, hospital, retail, restaurant, etc.) will determine the fidelity of prediction techniques required. For example, if a parking structure mainly serves a retail and restaurant district, it may be very easy to predict that customers arriving at the beginning of the day will stay about 8 hours for work while those arriving at other times will stay for 1-2 hours for lunch or shopping. In this case, a simple statistical regression of the stay time vs. arrival time may be sufficient to properly stage vehicles.

In other cases, more complex regression and machine learning techniques that can account for a wider variety of factors can be employed. Machine learning is a process by which a computer uses information about previous cases to predict future outcomes [55]. This encompasses simple regression techniques such as least-squares regression to more complex artificial neural networks and Gaussian Process models. Generally, the more non-linear a problem, the more complex of a technique required to adequately predict outcomes.

The problem at hand can be solved using a supervised learning approach where the inputs and responses are labeled by the user [55]. The primary response is the time that a customer will stay parked in the garage and/or the time when he or she will retrieve their car. Table 2.3 identifies potential features to be used to train the machine learning models.

Recent focus on machine learning throughout academia and industry has made fitting and implementing machine learning models relatively trivial (especially for well-structured and labeled data). Given that the data from all of Volley's garages is naturally stored and ordered, gathering the training set is also relatively simple. Hence, the real challenge is in how to leverage the predictions. Significant simulation studies will be conducted to properly size any queuing area or plan a staging strategy based on the performance of the model. The ultimate strategy could include preferring to keep equipment unoccupied near payloads that are nearing their retrieval times, unblocking cars that are predicted to be retrieved, to actually starting a retrieval and placing payloads into a holding area if the prediction models are precise.

Table 2.3: Features for Predicting Retrieval Times

Feature
• Time of day • Day of week • Parking structure type (e.g. residential or retail) • Parker type (e.g. daily or monthly) • Individual user ID (either license plate or a known user to the system)

2.7.2 Machine Learning for Customer Safety

Another application of machine learning to the present problem is to increase customer safety by identifying passengers, babies, pets, etc. remaining in the vehicle. It is very possible to forget a baby in the car or for a passenger to purposefully stay in the car to see the inside workings of the garage. While the passenger will be safe if they remain in the vehicle, preventing these cases is clearly preferable.

The first step to preventing this is to ask the user whether they have left anything in the car before each insert. Beyond this, combining visual and infrared cameras with machine learning could potentially catch a passenger or pet left behind. On the visual side, training the algorithm to search for movement in the bay that is supposed to be empty could alert the user to someone left behind. The algorithm could also be trained to identify items or objects that do not conform to the expected shape of the car. This could alert the user to items left accidentally on the roof of the car, which is a hazard to the automated system.

With the infrared system, the machine learning algorithm must be trained to detect heat signatures through the windows of the vehicle. This is necessary to attempt to identify a sleeping baby. This is a challenging classification problem since many different parts of the car are going to have infrared signatures from the hot engine to dark paint on a hot day. More investigation is required to determine if this is a feasible approach. Ultimately, helpful safeguards should be implemented, but some responsibility rests with the user.

2.7.3 Reclaimable Garage

Parking demand and regulations are very uncertain in today's world of changing car ownership practices, autonomous vehicles, and increasing mobility options. As such, investing in a 30 year structure dedicated to parking is not an attractive option to developers. Robotic parking inherently addresses this challenge by removing the need for ramps, so each level could be repurposed in a future with lower parking needs. Additional innovations can expand upon the ability to reclaim the space.

A challenge to reclaiming the space is that a livable space requires about 13 feet of ceiling height, whereas to save space, an automated parking garage may only need about 7 feet. This means that one could either save vertical space but limit the options for reclamation or build tall ceilings that reduce the benefit of automated parking. A third option combines the 2 concepts developing garage space with the full height but then installing temporary mezzanine structures on each floor. Thus when a floor is used for parking, the mezzanine level effectively splits the level to take advantage of the entire vertical space. By regulation, the mezzanine may not be able to cover the entire floor, however this allows the controller to

park shorter cars in the mezzanine area while reserving the full height spots for taller trucks and SUVs. Then, when the building owner decides to reclaim the space, the temporary mezzanine can be removed and the full height can be used once again.

2.7.4 Modular Garage Slots

Space flexibility provides an alternative to reclamation to future proof the garage structure. Traditional parking structures may be modified by repainting to add motorcycle parking, adding bike racks or electric vehicle charging spots, or allowing storage containers to park in their spaces. In general, this represents a small change to the parking structure and is not scalable to overhaul a traditional parking structure that no longer serves as many cars.

Volley's tray-based, automated parking solution provides a strong base to leverage product-family and modular design to continuously update the function of a "parking" structure. There are now a myriad of ever-changing mobility options and preferences including cars, motorcycles, bikes, scooters, electric vehicles, and walking. Transitioning a parking garage to a mobility hub requires the ability to quickly respond to changing demands. A promising approach to enable responsive change is modularity [56].

Modularity involves the design of swappable modules that can be seamlessly inserted into the system to change its function. More research is required to identify the baseline size; however, assuming the baseline tray is designed to accommodate an SUV, then the other mobility options could fit on smaller, more compartmentalized modules. Hence, instead of using the $17 \times 7\,\text{ft}$ tray to hold one car, a module can be installed that could store 4 motorcycles, an entire bike or scooter rack (with charging), 2 $7 \times 7\,\text{ft}$ storage units, or an electric vehicle charging station.

This modular concept can be expanded to add services to the garage as well. For example, a parking spot could be replaced with a car wash or oil change station. Because cars are inherently safe inside the parking structure, groceries or other packages could be delivered to unlocked cars or left on the car's tray for the customer. Becoming a mobility hub instead of simply a parking garage is a key differentiator that would enable Volley to grow into a true brand instead of just another parking garage.

2.8 Chapter Summary

This chapter has reviewed the improvements to the main controller software, mechanical systems, user experience, and system design practices that Volley is developing to overcome the gaps identified in Section 1.3. From discussions with many people throughout the industry, these challenges are the main drivers behind the failures of previous automated parking projects. These improvements combined with extensive testing throughout the design and implementation process will enable Volley to provide a more reliable, lower cost system then previous attempts. Further enhancements aimed at improving the customer experience, reducing costs, improving safety, and future-proofing the structure are also discussed. The following chapter discusses the implementation and testing plan in further detail.

Chapter 3

Integration & Testing Plan

This chapter provides an overview of the integration and testing plan and methodology Volley is implementing to evaluate the software and hardware system alternatives. As mentioned briefly in Section 2.6.4, Volley's systems engineering plan loosely follows the V Diagram. In a departure from strict systems engineering practices, requirements are not formally defined and flowed down through the process. Depending on how the product's complexity evolves, more formal requirements definition may be required in the future. This chapter begins by discussing the concept development and design phase. The majority of the chapter is then spent discussing the component and system-level testing before discussing the final system integration and validation process.

3.1 Concept Development

The concept development stage is where the overall makeup of the system is defined. High level questions, such as should the system be a rack-and-rail or AGV system, are asked and discussed during this phase. Initial looks at the trade-space are also conducted to identify necessary performance parameters (e.g. how fast should the AGV move, how many AGVs are required) to explore as the design is formalized.

The hardware concept that Volley has identified is the innovative, small AGV discussed in Section 2.4 moving payloads on trays with castors. This is an exciting concept because the order-of-magnitude cost reduction opens up many possibilities at the concept development phase. First, the cost is low enough where it is cost effective to place an AGV on every spot. In doing so, the system complexity is reduced and the theoretical response time can be greatly improved. Next, the small size, simplicity, and low-cost opens a wide variety of maintenance and system robustness opportunities. Because of the small physical size, the AGVs can be easily dragged by a human operator or another AGV if a breakdown occurs. Further, because the payloads are seated on castors, they can be moved manually if required. Finally, the simplicity of the AGV enables routine maintenance (e.g. tire change or motor replacement) to be handled by local technicians with limited training, while more complex overhaul can be completed at a depot with replacement AGVs easily shipped to the customer or kept in stock. This operations and maintenance concept drives the testing requirements defined in the following section.

3.2 Component Testing

A significant amount of effort will be spent at the bottom of the V diagram iterating between component design and testing. The primary components that require extensive testing are the AGV wheels, motors, and the tray's castors. The other sub-systems (lifts and bay) also could require component-level testing if they are produced in-house; otherwise, the components are tested by the manufacturer and, hence, would not need testing from Volley. A primary design philosophy is to use commercial off-the-shelf (COTS) parts throughout the design. This limits the need to test the function and quality of purchased parts as this is covered by the supplier. Thus, the primary component level tests to consider are mainly environmental and wear tests.

While the garage is a relatively closed and controlled system, it is not a very clean environment. The cars and payloads can bring any number of hazards in from the outside, including dirt, dust, water, snow, and oil. These substances all decrease the cleanliness of the garage floor and tray. In these cases, the performance of the castors and wheels need to be explored to determine the required mitigation policy (e.g. how often should the garage floor be cleaned). A test rig is developed to test the performance of the castors and wheels when encountering these hazards. Generally, the force required to move and swivel the castors is explored.

The wear of the castors and wheels is the main concern for the component-level testing. This has a direct impact on the maintainability (and therefore cost) of the system and should be well-understood before settling on a particular option. Drum test rigs are available to run the wheels and castors at the speed and force expected during implementation. As the trays will move only a handful of times each day, months or years of wear can be put on the castors in a relatively short amount of time. The testing will include both continuous movement and cycles to identify different failure points of each. A similar test rig that can resist the motor's movement is also leveraged to explore the continuous and cycle life of the AGV's motor.

The goal of these tests is to identify the components that have the best combination of performance, cost, and reliability. Understanding the longevity of each component will drive the maintenance requirements, which also drives a variety of choices about how to handle failures, how to staff the garages, and the types of maintenance contracts/guarantees to offer. With these wear components selected, the next step is to perform sub-system testing to ensure that the AGVs, lifts, and bays meet their expected performance levels.

3.3 Sub-system Testing

The next level involves testing each of the sub-systems, which are the lifts, bays, and AGVs/trays. Similar to the component-level testing, each sub-system is tested against each of its functional requirements. These will be further defined as the system matures; however, Table 3.1 defines an initial set of functions for each sub-system.

The individual functions for each sub-system are tested to examine whether the sub-systems meet their requirements. The purpose of sub-system testing is to define independent tests for each function to identify areas that require further design. For example, if the AGV is not able to rotate the payload, then the AGV may need to be re-designed or the function can be allocated to another sub-system (e.g. a turntable). Thus, the sub-system testing is intended to identify issues where the sub-systems are not able to provide the performance required by the overall system. While examining the performance of each sub-system, the reliability and life of each is also tested through wear and extreme environment testing

Table 3.1: Sub-system Functional Breakdown

Sub-system	Functions
AGV	• Move payload along line • Interface with payload • Rotate payload • Change payload movement direction • Automatically locate within the grid • Identify lift is in place before proceeding
Lift	• Move payloads between garage floors • Locate and interface at each floor • Maintain flatness
Bay	• Manage insert and retrieval operations • Ensure customers are clear of the bay before proceeding • Identify the dimensions of a payload

(similar to the component testing). Following the sub-system testing, complete system-level testing is discussed in the next section.

3.4 System-level Integration Testing

System-level testing takes the sub-systems with verified performance and integrates them into a mock-up of the final system. At this point, the interfaces between the sub-systems (e.g. communication between the controller and equipment, interface and safety check between AGV and lift, etc.) are implemented and tested. The performance and robustness of these interfaces are key to the overall system's success, so significant amount of stress testing is conducted.

In parallel, the verified performance parameters are passed to the simulation to check the system's performance in the digital world. Simulation should be used throughout to identify any incorrectly defined requirements, and this represents a final check. For example, now that the sub-systems' performance parameters are verified, the AGV speed in the simulation is updated to the actual speeds seen in during the verification test. This is done for all of the functional parameters tested throughout the testing cycle. Thus, with these parameters dialed in, the simulation is run to see if the overall metrics (e.g. customer wait time) are still met. If there are problems, then either the requirement must be loosened or some of the lower-level functional requirements must be tightened. Assuming there are no problems with this verification process, the final step is full-scale system validation.

3.5 System Validation

Once the system verification process is completed, there is a high-degree of certainty that the designed system can achieve the requirements set forth. Thus, this final step involves full-scale implementation and testing against the concept of operations. For Volley, this

includes installation in a full-scale test facility or into an initial project with ample time to test the implemented system.

The beauty of the automated garage system is that the implementation testing is relatively straightforward. Insert and retrieval requests can be triggered according to the expected demand pattern. Then, empty trays can be moved through the garage as they would with payloads. For added realism, the trays could be weighted to get better information on the wear and battery usage during operation. In this way, all aspects of the system, aside from the user interaction, can be tested to validate that the system can provide the services laid out in the concept of operations. This testing is conducted over an extended period of time to ensure that the garage is capable of continuous operation. Once Volley is comfortable with the performance and continuity of operations, the garage is opened for customers.

Chapter 4

Conclusion

Autonomous parking garages are becoming increasingly necessary as the density in cities continues to rise. The primary benefit of autonomous parking is that by removing the human from the vehicle, cars can be parked very close together and can block other customer vehicles. Further, by removing the need for customers to walk in the garage, space required for customer-focused elevators, stairs, and ventilation systems is freed for more parking. Overall, an autonomous system can approximately halve the space required for each parking space.

Of the potential solutions, an autonomous system leveraging AGVs and vertical lifts to park vehicles and payloads on trays in the garage area is a promising solution. Compared to rack-and-rail systems, the AGV system can accommodate oddly shaped buildings more naturally and offer expanded redundancy to remove cars from the garage. Further, by leveraging the high ceiling and mezzanine concept and/or the modular design approach discussed in Section 2.7.3, the AGV-based system can better adapt to future changes in mobility preferences. Thus, Volley has chosen to pursue the AGV-based system.

Volley began the design process by identifying the shortcomings of previous systems. These gaps are:

1. Software-based planning and routing algorithm deficiencies
2. Excessive complexity and customization of mechanical components
3. Limited consideration for user experience or customer interactions
4. Poor systems-level design and analysis

The shortcomings are then used to inform design decisions made in an effort to improve against the system evaluation criteria. These system goals are:

- Minimize structure size/Maximize density
- Minimize customer wait time
- Maximize throughput
- Minimize operational cost

Chapter 2 discusses the proposed design changes that address the identified problems. Instead of forcing a warehousing solution onto the problem, Volley is designing the controller from the ground up to be robust in that deadlocks are prevented but alternative routes out of the garage are naturally identified. For the AGV, a new locomotion concept that splits the load support function from the movement function enables significant simplification of the AGV. This order-of-magnitude cost reduction can open many other opportunities and control concepts than previous system (e.g. AGVs under every tray, less costly maintenance plans, etc.). The user experience is also planned to include options to pay for different pricing tiers and the inclusion of a web app to schedule pickups and handle payment. Finally, the overall project design process is updated and automated to help the developer understand the variety of trade-offs available to select the layout that works for his or her particular project.

These improvements throughout the system are then tested as described in Chapter 3. The integration and testing process is loosely based on the systems engineering V Diagram, where system requirements are flowed from systems to sub-systems to components. Upon each step of implementation, the designed component or system is tested against its requirements to verify performance. Upon verifying the system through detailed simulation and a scale model, the system is integrated into a full-scale test facility or initial project with extended time to integration test.

The next steps are to continue development of the controller, AGV, and system design suite. The AGV in particular represents a promising step forward that requires refinement and productization before implementation into a garage. Ultimately, the potential benefits from the 10x cost reduction will open new opportunities for Volley garages.

Bibliography

[1] Anthony P Chrest, Mary S Smith, Sam Bhuyan, Mohammad Iqbal, and Donald R Monahan. *Parking structures: planning, design, construction, maintenance and repair*. Springer Science & Business Media, 2012.

[2] Todd Litman. *Parking management best practices*. Routledge, 2018.

[3] Bud Ovrom. The parking dilemma: innovative solutions for parking and parking requirements. In *Automated Parking Conference*, 2011.

[4] Don Monahan. Man vs machine: Is robotic parking right for your project? In *International Parking Institute*, 2012.

[5] Donald C Shoup. The trouble with minimum parking requirements. *Transportation Research Part A: Policy and Practice*, 33(7-8):549–574, 1999.

[6] Robert Crommelin and Inc. Associates. Entrance-exit design and control for and major parking and facilities. In *"SEMINAR '72" Los Angeles Parking Association*, 1972.

[7] Malcolm Douglass and Steve Abley. *Trips and parking related to land use*. Number 453. 2011.

[8] Gary Cudney. Parking structure cost outlook for 2017. Technical Report and, Carl Walker, 2017.

[9] Dennis FX Mathaisel, Joel M Manary, and Ned H Criscimagna. *Engineering for sustainability*. CRC Press, 2012.

[10] Alexei Barrionuevo. Hate valet? not to worry; help is at hand, 2013. URL `https://www.nytimes.com/2013/03/03/realestate/automated-parking-garages-for-the-car-obsessed.html?module=inline`.

[11] Frances Robles. Road to robotic parking is littered with faulty projects. *The New York Times*, 27, 2015. URL `https://www.nytimes.com/2015/11/28/us/road-to-robotic-parking-islittered-with-faulty-projects.html?_r=0`.

[12] Jonathan Miller. When a garage goes wrong, 2005. URL `https://www.nytimes.com/2005/10/30/nyregion/when-a-garage-goes-wrong.html`.

[13] Dennis J.L. Siedlak, Todd M. Schmidt, Olivia J. Pinon, and Dimitri N. Mavris. A methodology for the parametric exploration of the impact of production planning on the early stages of design. In *Proceedings of ASME 2014 International Manufacturing Science and Engineering Conference*, number MSEC2014-3974. American Society of Mechanical Engineers, June 2014.

[14] Dennis JL Siedlak, Paul R Schlais, Olivia J Pinon, and Dimitri N Mavris. Supporting affordability-based design decisions in the presence of demand variability. In *Proceedings of ASME 2015 International Manufacturing Science and Engineering Conference*, number MSEC2015-9422. American Society of Mechanical Engineers, June 2015.

[15] Dennis JL Siedlak, Olivia J Pinon, Paul R Schlais, Todd M Schmidt, and Dimitri N Mavris. A digital thread approach to support manufacturing-influenced conceptual aircraft design. *Research in Engineering Design*, 29(2):285–308, 2018.

[16] A. R. Ismail and T. C. Melewar. A structural model to examine the antecedents and consequences of customer with experiential brands. The International Conference on Brand Management, 2010.

[17] Francis Buttle. Servqual: review, critique, research agenda. *European Journal of marketing*, 30(1):8–32, 1996.

[18] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[19] Walter Hoffman and Richard Pavley. A method for the solution of the n th best path problem. *Journal of the ACM (JACM)*, 6(4):506–514, 1959.

[20] Xiaojun Zhou, Lifeng Xi, and Jay Lee. Reliability-centered predictive maintenance scheduling for a continuously monitored system subject to degradation. *Reliability Engineering & System Safety*, 92(4):530–534, 2007.

[21] Basil Halperin, Benjamin Ho, John List, Ian Muir, et al. Toward an understanding of the economics of apologies: evidence from a large-scale natural field experiment. Technical report, The Field Experiments Website, 2018.

[22] Wei Chen, Janet K Allen, Kwok-Leung Tsui, and Farrokh Mistree. A procedure for robust design: minimizing variations caused by noise factors and control factors. *Journal of mechanical design*, 118(4):478–485, 1996.

[23] Dennis J.L. Siedlak, Olivia J. Pinon, Bradford E. Robertson, and Dimitri N. Mavris. Robust simulation-based scheduling methodology to reduce the impact of manual installation tasks on low-volume aerospace production flows". *In Press for the Journal of Manufacturing Systems*, 2017.

[24] Jose M. Framinan, Rainer Leisten, and Rubén Ruiz García. *Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools*. Springer, 2014.

[25] Tapan P Bagchi. *Multiobjective scheduling by genetic algorithms*. Springer Science & Business Media, 1999.

[26] Jasbir Arora. *Introduction to optimum design*. Academic Press, 2004.

[27] John W. Fowler and Oliver Rose. Grand challenges in modeling and simulation of complex manufacturing systems. *SIMULATION*, 80(9):469–476, 2004. doi: 10.1177/0037549704044324. URL `http://sim.sagepub.com/content/80/9/469.abstract`.

[28] Dennis J.L. Siedlak. *Robust Scheduling Methodology to Reduce Risk in Aerospace Production Systems*. PhD thesis, Georgia Institute of Technology, 2016.

[29] Gary Stump, Timothy W. Simpson, Mike Yukish, and Lorri Bennett. Multidimensional visualization and its application to a design by shopping paradigm. In *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, number AIAA-2002-5622, Atlanta, GA, USA, 4-6 September 2002.

[30] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Optimization*, volume 7 of *Wiley Series in Engineering Desifn and Automation*. John Wiley & Sons, 2000.

[31] Thomas Morton and David W Pentico. *Heuristic scheduling systems: with applications to production systems and project management*, volume 3. John Wiley & Sons, 1993.

[32] Fred Glover and Gary A Kochenberger. *Handbook of metaheuristics*. Springer Science & Business Media, 2003.

[33] Garret N. Vanderplaats. *Multidiscipline Design Optimization*. Vanderplaats Research & Development, Inc., Monterey, CA, 1st edition, 2007. ISBN 0944956041.

[34] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[35] J. Wood, H. Wright, and K. Brodie. Collaborative visualization. In *Visualization '97., Proceedings*, pages 253–259, Oct 1997. doi: 10.1109/VISUAL.1997.663890.

[36] John Stark. *Product Lifecycle Management: 21st Century Paradigm for Product Realisation*. Springer, 2005.

[37] John M. Usher, Utpal Roy, and Hamid R. Parsaei. *Integrated Product and Process Develoment: Methods, Tools, and Techologies*. John Wiley & Sons Inc., 1998.

[38] Jeffrey Heer, Frank Van Ham, Sheelagh Carpendale, Chris Weaver, and Petra Isenberg. Creation and collaboration: Engaging new audiences for information visualization. In *Information Visualization*, pages 92–133. Springer, 2008.

[39] M. Tobiasz, P. Isenberg, and S. Carpendale. Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1065–1072, Nov 2009. ISSN 1077-2626. doi: 10.1109/TVCG.2009.162.

[40] James J Thomas. *Illuminating the path: the research and development agenda for visual analytics*. IEEE Computer Society, 2005.

[41] Daniel A Keim, Gennady Andrienko, Jean-Daniel Fekete, Carsten Görg, Jörn Kohlhammer, and Guy Melançon. *Information Visualization*, volume 4950 of *Lectures Notes in Computer Science*, chapter Visual Analytics: Definition, Process, and Challenges, pages 154–175. Springer Berlin / Heidelberg, 2008. doi: 10.1007/978-3-540-70956-5.

[42] Daniel A Keim, Florian Mansmann, and Jim Thomas. Visual analytics: How much visualization and how much analytics. *SigKDD Explorations Journal*, December 2009.

[43] Iiro Harjunkoski. Deploying scheduling solutions in an industrial environment. *Computers & Chemical Engineering*, 91:127–135, 2016.

[44] Mike Yukish, Gary M Stump, and Sara Lego. Visual steering and trade space exploration. In *2007 IEEE Aerospace Conference*, pages 1–9. IEEE, 2007.

[45] Julian Heinrich and Daniel Weiskopf. State of the art of parallel coordinates. *STAR Proceedings of Eurographics*, 2013:95–116, 2013.

[46] Daniel A Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. Visual analytics: Scope and challenges. In *Visual data mining*, pages 76–90. Springer, 2008.

[47] Christopher Ligetti, Timothy W. Simpson, Mary Frecker, Russell R. Barton, and Gary Stump. Assessing the impact of graphical design interfaces on design efficiency and effectiveness. *Journal of Computing and Information Science in Engineering*, 3(2): 144–154, June 2003. doi: 10.1115/1.1583757.

[48] D. N. Mavris, O. J Pinon, and D. Fullmer Jr. Systems design and modeling: A visual analytics approach. In *27th Congress of International Council of the Aeronautical Sciences (ICAS)*, 2010.

[49] Karel Mařík, Zdenek Schindler, and Petr Stluka. Decision support tools for advanced energy management. *Energy*, 33(6):858 – 873, 2008. ISSN 0360-5442. doi: http://dx.doi.org/10.1016/j.energy.2007.12.004. URL `http://www.sciencedirect.com/science/article/pii/S0360544207002216`. {PRES} 2006. 9th Conference of Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction - {PRES} 2006PRES 2006.

[50] Mark Velasquez and Patrick T Hester. An analysis of multi-criteria decision making methods. *International Journal of Operations Research*, 10(2):56–66, 2013. URL `http://orstw.org.tw/ijor/vol10no2/ijor_vol10_no2_p56_p66.pdf`.

[51] Evangelos Triantaphyllou. *Multi-criteria decision making methods: a comparative study*, volume 44. Springer Science & Business Media, 2013.

[52] Serafim Opricovic and Gwo-Hshiung Tzeng. Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS. *European Journal of Operational Research*, 156(2):445 – 455, 2004. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/S0377-2217(03)00020-1. URL `http://www.sciencedirect.com/science/article/pii/S0377221703000201`.

[53] Alexander Kossiakoff, William N Sweet, Samuel J Seymour, and Steven M Biemer. *Systems engineering principles and practice*, volume 83. John Wiley & Sons, 2011.

[54] Elie Allouis, R. Blake, S. Gunes-Lasnet, T. Jorden, B. Maddison, H. Schroeven-Deceuninck, M. Stuttard, P. Truss, K. Ward, R. Ward, and Mark Woods. A facility for the verification & validation of robotics & autonomy for planetary exploration. 05 2013.

[55] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160:3–24, 2007.

[56] Seth Libby, Dennis J Siedlak, Heriberto D Solano, Olivia J Pinon-Fischer, and Dimitri N Mavris. Cost-capability analysis of uas family and flexible factory design. In *17th AIAA aviation technology, integration, and operations conference*, page 4250, 2017.

