


Profitability of Proposed Grocery Store Layouts Using Simulation Methods



Abstract

Before a grocery store opens key operational decisions must be made with no historical data. One important decision is how to optimally lay out the store to maximize consumer spending. This work reviews existing literature on simulation to optimize grocery store layout, uses computer vision techniques to transform a store diagram into a digital representation, and applies simulation methods to approximate which of the layouts proposed by a store designer would result in the highest amount of impulse purchasing. Output analysis methods are used to compare these results to determine whether one design outperforms the others.

Background

Overview

This topic was chosen after connecting with a local food co-op planning to open in my community [14]. We discussed their current status, and what aspects in the store opening process might be assisted with a simulation study. The Store Development Committee had been working with a store designer on planning the layout of the store and suggested that it could be useful to have an analysis of which of the proposed floorplans is likely to be most profitable. At this stage, the floorplan is defined as the block layout and the assigned department label. It is assumed that the floorplan is designed by someone with knowledge of the planned inventory and available space. An example can be seen in Figure 1. Due to the request to keep planning materials confidential, this report demonstrates the same simulation and analysis methods applied to fictional, example layouts.

Literature Review

Conventional grocery stores typically have very low profit margins, averaging about 2.2% in the US [2]. Although there is less data available on the profit margins of food co-ops, the management team expects similarly slim profit margins. In order to maintain profitability, several decisions can be made to decrease costs or increase sales. I focus on increasing sales by choosing a store layout to increase customer impulse purchases. Impulse purchases are unplanned purchases, and are viewed as an opportunity to increase sales over what a customer was planning to spend on commodities through factors the store can control. Impulse purchases are said to account for 30-50% of sales, and 3 out of 4 shoppers report making some purchasing decisions in-store [8, 12]. Hoch and Loewenstein suggest that physical proximity is the strongest inducer of reference-point shifts, which are sudden desires that result in unplanned behavior [6]. Therefore, a longer path through the store is desirable as it provides more opportunities for stimulating additional impulse purchases. Since the store is not yet open and detailed product placement data does not exist, I will use average customer path length as a proxy for impulse shopping. This relationship is well-established in the literature.

There are many approaches to simulate the path a customer will take within a grocery store. Larson uses a proprietary PathTracker® system which affixes RFID tags to grocery carts to study common paths [9]. The paths identified are specific to the store layout where the data was collected, and cannot be applied to other layouts arbitrarily. Hui, Fader, and Bradlow note the similarity of grocery store shopping to the traveling salesman problem (TSP) and study the ways customer paths systematically differ from the optimal TSP path [7]. They found that customers deviate from the optimal path on average by 28% (ranging from 5 to 95%). The first type of deviation, order deviation, is when customers do not visit the next-closest item to where they currently are. This averaged 3% of the deviation. The second type of deviation, travel deviation, is when customers visit the next-closest item, but take a path other than the shortest path to get there. This averaged 69% of the variation. Order deviation is strongly correlated to basket size, while travel deviation is uncorrelated to basket size. For this reason, in this simulation for the purpose of estimating additional purchases only order deviation will be considered. Several papers use a one-step-ahead approach in which a customer selects the next item from their list with a probability inversely proportional to its distance from their current location [1, 3]. That approach will be adapted for this simulation.

Additional approaches exist for optimizing grocery-store layout to stimulate impulse shopping. Bhadury, Batta, Dorismond, Peng, and Sadhale apply the p-dispersion model to spread common “must-have” items across the store, thereby maximizing the customer’s exposure to impulse items [1]. Significant improvement was achieved using this method, but the scope and problem definition differ from that of selection from a fixed set of store layouts at the department-level. Ozgormus and Smith propose a method for increasing revenue while considering adjacency preferences and validate it on two locations of the Turkish grocery chain Migros [10]. Again, large improvements were achieved (3-4% increase in revenue), but our problem definition varies from the one described here; in [10] the layout is fixed and only the contents of those fixed departments vary. Finally, Dorismond outlines a simulation-based tool to guide periodic changes in a supermarket layout and an approach to dynamically position promotional products in a retail store [3]. There are several interesting findings, but again the block layout is assumed to be fixed and the customer path and shelf allocation are explored. Although none of these solutions exactly apply to the problem of comparing several proposed layouts to determine which maximizes customer path length, several of the strategies used in those simulations can be applied.

Method

Parsing the Store Diagram

The first task in simulating the customer flow and purchasing activity in a store is translating a document representing the store’s layout into a representation that can be used by the simulation. A simplified example of a floor layout plan is shown in Figure 1. There is a single column which contains a legend mapping colors to the name of the product type. The location where items of that type are stored is entirely shaded in the same color. There may be some additional features of the diagram, such as dimensions, bathrooms, and more, which are ignored.

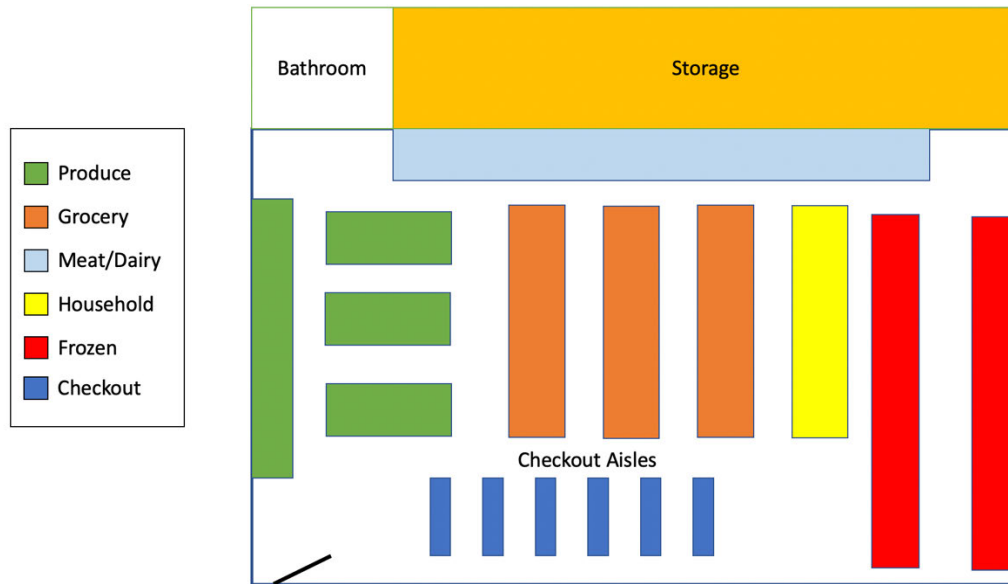


Figure 1: Example grocery store layout

Store Legend Colormap

To be able to map regions in the store to items a simulated customer is interested in, the legend should be parsed to map an RGB value to a department name. This is achieved as follows:

1. The legend is saved as a separate image, containing one column of color blocks matched with the aisle name per line
2. Use Tesseract OCR (Optical Character Recognition) to read all text from the image
3. Split on the newline delimiter '\n' and automatically clean up special characters. Results of OCR are typically not perfect, and manual cleanup may be required as well.
4. Identify unique colors in the legend image which appear in more than .05% of the pixels in the legend. KMeans clustering algorithm is used since most colors have variations in the pixel level RGB values.
5. Sort the colors in the order they appear in the image, top to bottom, and assign them to the department names.

There may be more optimal methods for achieving these tasks automatically and with lower error, but given the focus of the course further discussion is out of the scope of the project. This approach assumes black, white, and gray are not used as department colors and no colors are used in large amounts in the legend besides department colors. High and low ranges are specified as well to provide a flexible threshold for images where the borders have some blending. The output of this for the example layout would be a table as follows:

Table 1: Example legend table

Department	BGR_Centroid	Low_BGR	High_BGR	RGB
produce	[84, 170, 125]	[69, 155, 110]	[99, 185, 140]	[125, 170, 84]
grocery	[67, 129, 223]	[52, 114, 208]	[82, 144, 238]	[223, 129, 67]

meatdairy	[234, 174, 75]	[219, 159, 60]	[249, 189, 90]	[75, 174, 234]
household	[84, 253, 255]	[69, 238, 240]	[99, 268, 270]	[255, 253, 84]
frozen	[35, 50, 234]	[20, 35, 219]	[50, 65, 249]	[234, 50, 35]

BGR values are used primarily in the OpenCV package which is being used in this code. With this mapping sections of the store layout map can be assigned to different departments.

Preprocess Store Layout

The first step for the store layout is also to save it as a separate image, containing only the floorplan that a customer could walk through. I resize each image to 500x500 pixels to standardize maximum path size and reduce the search space. Since items can only be placed on the perimeter of any aisles, it's necessary to find the coordinates of the perimeter for all departments.

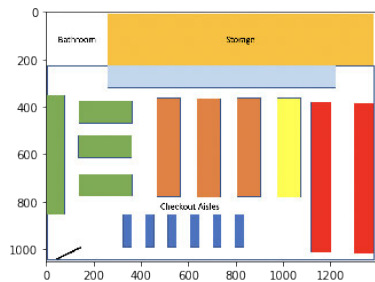


Figure 2: Original Image.



Figure 3: Produce section pixel mask.

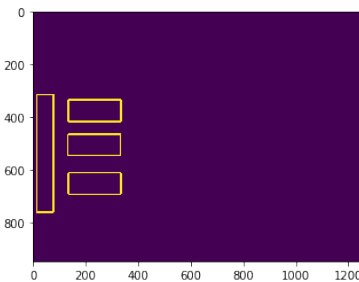


Figure 4: Perimeter where products are placed

This sequence shows edge detection for the produce section. First, the color mask created from the high and low values for any department in the legend is applied to filter out pixels that are not part of the section, resulting in Figure 3. Then, a basic edge detection algorithm is applied to identify the perimeters, shown in Figure 4. The pixel indices are stored as options for item placement for each department. A buffer of 10 pixels is set to indicate that products cannot be placed on the perimeter of the store. For example, the leftmost produce section will not have products on its leftmost perimeter, because that is against a wall and inaccessible to the customer.

Finally, a combined mask of all departments is created to denote where customers can and cannot walk for the path generation algorithm. This is shown below.

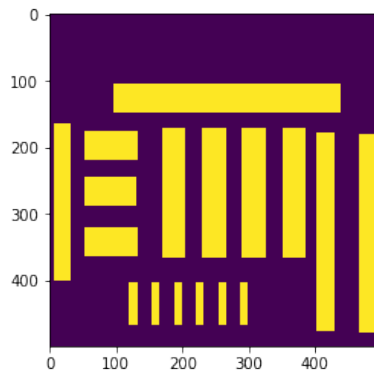


Figure 5: Walking path mask, purple denotes where customers can walk, yellow denotes obstacles

Create Customer Shopping Lists

There are a few options to create customer shopping lists. One option is to create customer profiles based on clustering analysis, and create shopping lists for each profile [1, 9]. Another is to use association rule mining to identify rules for products customers typically buy together and create lists based on those rules [10]. The simplest option, which I will use here, is to directly use customer purchase lists. This limits the flexibility of the customers and the variability introduced into the simulation, which now mostly comes from path choice. This limitation is surpassed by using a large dataset. A more complex customer list creation mechanism would be a great future step to explore, especially as more information about expected customers becomes apparent since the profiles or rules can be tweaked to be more realistic.

The dataset I used is the Instacart Online Grocery Shopping Dataset 2017 [13]. At the time I tried to access it the website was temporarily unavailable, so the data was obtained through Kaggle [10]. The dataset was chosen because of the large number of orders (around 3 million), the pre-existing categorization into departments, and, most importantly, because each order does not reflect impulse buys that were triggered by physical proximity. Some amount of impulse buying may occur due to recommender systems through the web app, and it is unclear whether that was a feature of the site prior to 2017. However, the Instacart orders should reflect a customer's "must-have" purchases more accurately than in-store data, which would also reflect impulse buys.

To apply the Instacart dataset to a different categorization, a mapping must be created between the departments in the dataset and those in the proposed layout legend. For the example grocery store layout in Figure 1, it is shown below:

Table 2: Mapping from example legend categories to Instacart Dataset categories

Figure 1 Department	Instacart Departments
Produce	Produce
Grocery	Bakery, International, Beverages, Dry Goods Pasta, Bulk, Pantry, Breakfast, Canned Goods, Snacks
Meat/Dairy	Meat Seafood, Dairy Eggs, Deli
Household	Other, Pets, Personal Care, Household, Babies
Frozen	Frozen
None	Alcohol, Missing

This is represented numerically in a table. Products in departments mapped to "None" are removed from orders. These department mappings are extended to product and order mappings, and to generate a customer grocery list we can simply query an order from the Instacart dataset with the remapped columns to apply to the grocery store layout. The training set used from Kaggle contains 131,209 unique shopping lists. An example list is shown below:

Table 3: One of the Instacart shopping lists with remapped departments

order_id	product_id	product_name	mapped_dpt	mapped_dpt_name
1260810	39275	Organic Blueberries	0	produce
1260810	37158	Chicken & Maple Breakfast Sausage	4	frozen
1260810	10957	Fridge Pack Cola	1	grocery

1260810	32650	White Giant Paper Towel Rolls	3	household
1260810	46175	Half Baked® Ice Cream	4	frozen
1260810	20682	Cinnamon Raisin Swirl Pre-Sliced Bagels	1	grocery
1260810	22571	100% Whole Wheat Cinnamon with Raisins	1	grocery
1260810	29095	Original Cream Cheese	2	meatdairy
1260810	35461	Blueberry Beet & Brown Rice Cakes	3	household
1260810	22289	Brownie Batter Core	4	frozen
1260810	43460	Apple, Juicy Red, Family Pack	0	produce
1260810	25999	Organic White Cheddar Macaroni & Cheese	1	grocery

The checkout section is always added to a customer shopping list as well, and it is always reserved to be the last item.

Translate Shopping List to Customer Path

Since item location within a given department is unknown, it is treated as a random location within the department. This means even if Customer₁ and Customer₂ have the same product on their shopping list they may find it in different locations within a proposed layout Layout₁. Similarly, if Customer₁ goes through Layout₁ twice, she may find the same item at different locations. In this way, there is variation introduced into the layout so no assumptions are made about where a product is placed. The coordinates, or pixel indices, for each product on a customer's shopping list are chosen randomly from the perimeter of the associated department in the store layout. Two examples of generating coordinates from the same grocery list are shown below, where item placement coordinates are shown in purple:

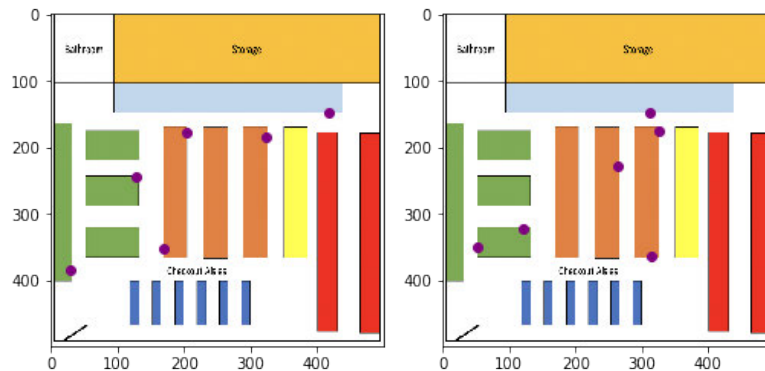


Figure 6: Two instances of the placement of products from the same underlying grocery list

After the item coordinates are determined, they must be ordered by when the customer visits each. Starting coordinates must be specified for the path generator. In this case it's the bottom left corner where the doorway is shown. From there, the approach in [4] is used. A diagram is shown below:

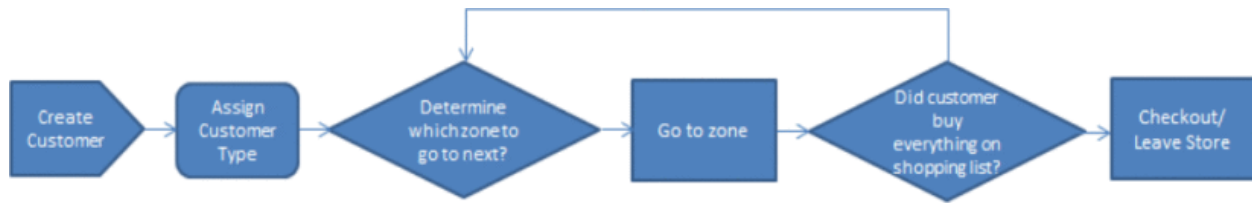


Figure 7: Diagram from [4] on one-step-ahead path generation

At this point, the customer has been created and the customer's shopping list has been assigned. The decision for which zone to visit next is determined by calculating the Euclidean distance between the current location and all future points to visit. The Euclidean distance is chosen because it is quick computationally and gives an approximation for the customer's travel distance. The probability vector \mathbf{P} corresponding to the unvisited points \mathbf{C} from current point $\mathbf{c} = [c_x, c_y]$ is computed as follows:

$\mathbf{D} = |\mathbf{C} - \mathbf{c}|$ \mathbf{D} is the distance from the current point to each point

$\mathbf{P} = (1/\mathbf{D})^5$ The probability is inversely proportional to the distance

$\mathbf{P} = \mathbf{P}/\sum \mathbf{p}_i$ Probability vector is normalized to sum to 1

This was selected based on the maximum distance of 707 for one step with the image resized to 500 by 500 pixels, and could be formulated differently depending on desired behavior. Based on those probabilities, the next point is selected and the current point is removed from the list of points to visit. Once the shopping list is empty, a random checkout line is selected and the customer moves there to leave the store. The results of ordering a coordinate set is shown for an example set of points below:

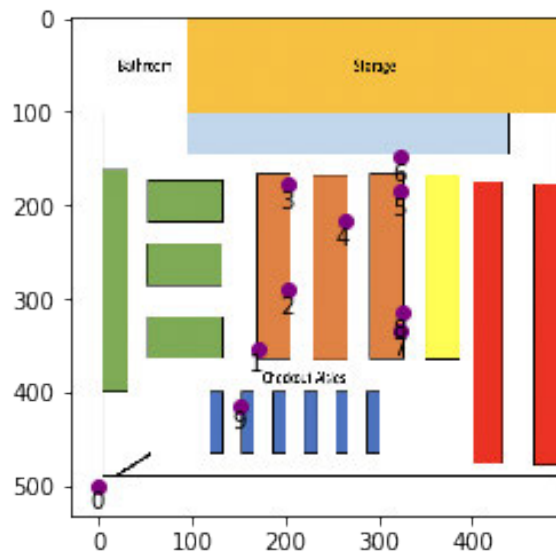


Figure 8: Customer visit locations denoted in purple, overlayed with assigned order

The last step of path generation is to actually generate the path between these ordered points. One approach is to use the Euclidean distance, and ignore the physical obstruction of the aisles. This is used in [1] and others. Another approach is to use a path generation algorithm. I'll use the A* algorithm, a common successor to Dijkstra's algorithm which uses heuristics to achieve high performance in less time [5]. However, it's still far less efficient than the simple computation of the Euclidean distance, and using the A* algorithm comes with a noticeable slowdown. For my implementation I track both Euclidean distance and A* path distance, with

the option to disable A* computation for faster results. Figure 9 shows the same set of ordered coordinates to be visited, and the path the customer takes using A* as compared to the shortest Euclidean path.

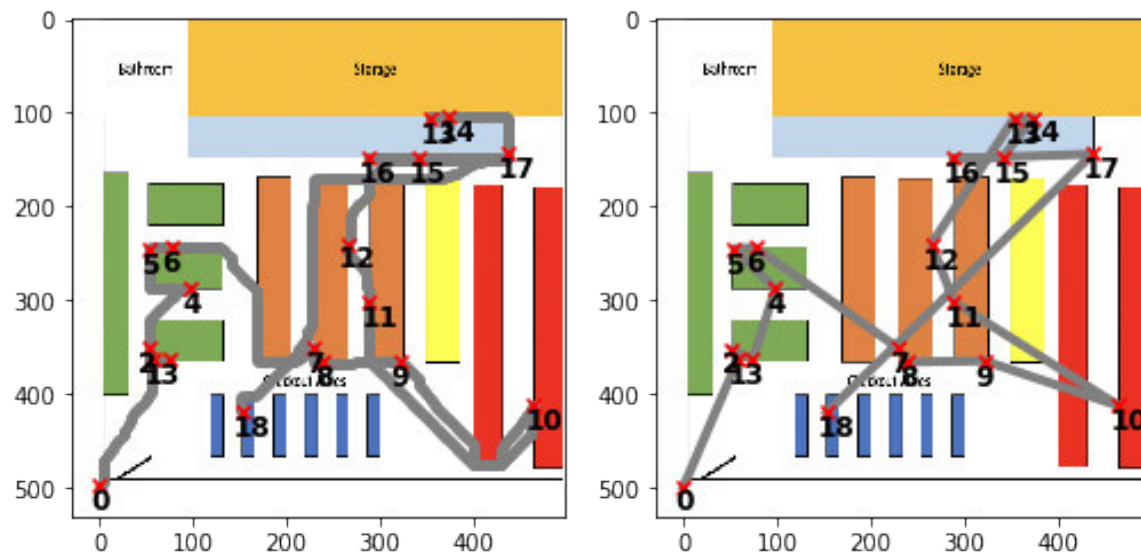


Figure 9: Customer path with A* shown on the left, customer path using Euclidean distance on the right

Summary

With all of these components together, there is now a way to simulate customer path distance through a store layout. The legend is parsed from an image to identify the department names matching each color. A store map is parsed from an image to represent physical obstructions and the outlines of each department are stored for potential product placement. Customer shopping lists are generated using the Instacart Grocery Online Shopping Dataset of 2017, which is remapped to the departments existing in the store layout plan. Finally, a customer's path is generated by randomly assigning locations within a department perimeter to each product, ordering the visitation of those locations using a one-step-ahead method, and measuring the distance between points with either the Euclidean distance or the length of the path generated by the A* algorithm.

Results

Other layouts must be compared in order to obtain results on the original task of choosing from several proposed layouts. Figure 10 below shows the three layouts created to compare between. In all of them the legend remains the same. To evaluate each layout, 1,000 customers are created, and each individual's path is tracked for the same grocery list through all of the layouts. The path distance is output. Output analysis techniques will be used to compare the layouts and determine whether one is substantially better than the others in terms of mean customer path.

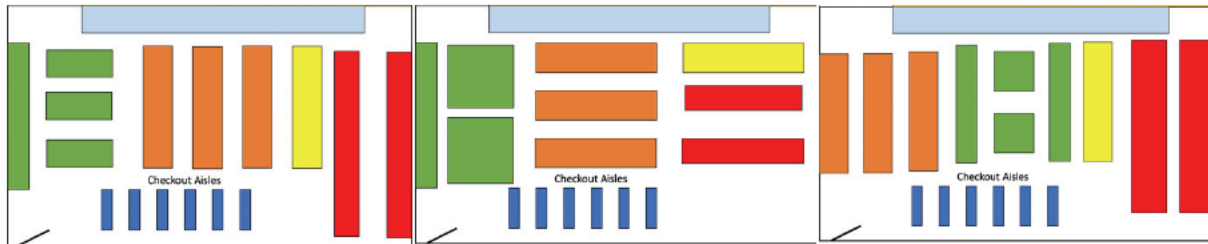


Figure 10: Three proposed layouts for a grocery store. The figure on the left, Layout 1, is the one used throughout the report. In Layout 2 in the center, many of the aisles are rotated to a horizontal orientation and the produce department shape slightly changes. On the right in Layout 3 the position of the grocery aisles switches with the produce section, and the produce section layout is altered.

The results are shown in Table 4 below. Using the Euclidean path distance, Layout 1 appears to generate the longest average customer path. Using the A* path distance, Layout 3 appears to generate the longest average customer path, with Layout 1 second.

Table 4: Average distance over 1000 paired samples of customer walking paths

	Euclidean path distance (pixels)	A* path distance (pixels)
Layout 1	1245.85	1859.86
Layout 2	1206.87	1664.74
Layout 3	1205.30	1881.03

Figure 11 shows that the values of each distance metric for each layout are approximately normal.

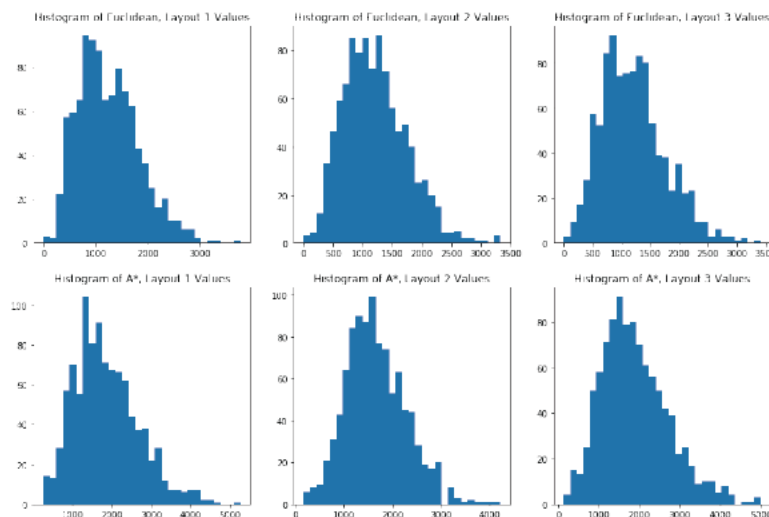


Figure 11: Images of histograms for each distance metric and layout over the 1000 customers

Since these samples were taken in a way that introduces dependence (the same customer moves through all three layouts), we can use paired confidence intervals to evaluate the difference between the means of each system. Results for the A* walking path are shown in Figure 12. There is a distinguishable difference between Layouts 1 and 2. Since the confidence interval with $\alpha=.05$ lies entirely above 0, it's been shown with 95% confidence that Layout 1 generates a longer average walking path than Layout 2. The confidence interval comparing Layout 1 to Layout 3 includes 0, so we cannot conclude that one layout is significantly better

than the other, although in this sample Layout 3 generated a higher average walking path. Finally, the confidence interval comparing Layout 2 to Layout 3 is entirely below 0, so it's been shown with 95% confidence that Layout 3 generates a longer walking path than Layout 2.

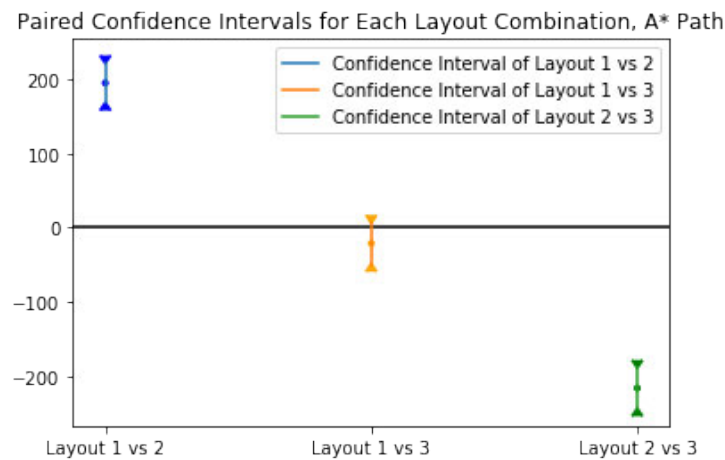


Figure 12: Results of comparing layouts pairwise by A* walking path

This analysis is repeated using the Euclidean path measure and results are shown in Figure 13. In this analysis, Layout 1 is shown to be better than Layouts 2 and 3 with 95% confidence, and Layout 2 and 3 are indistinguishable. This result differs from that found with the A* path, which is more representative of a customer's true walking path. This is an important finding because other papers evaluate their layout with the Euclidean distance, which is now shown to not always correspond to the walking path when accounting for obstacles.

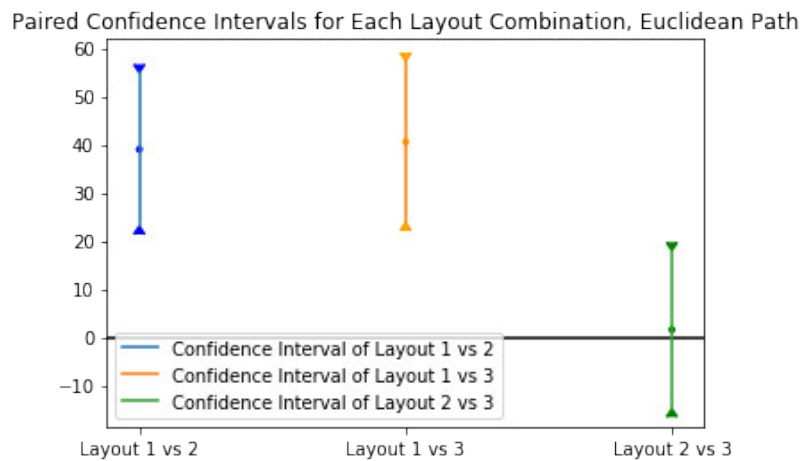


Figure 13: Results of comparing layouts pairwise by Euclidean walking path

Another way to compare many layouts is to use a selection method adopting the indifference-zone approach. This means comparing the best system, or subset of systems within the same performance zone such that we are indifferent between those systems. This method is not used here, but implementation would be a useful future step.

In addition, a brief timing analysis was completed. The time to compute one customer's path through each of the 3 layouts using both the Euclidean distance and the A* path distance averaged 37.8 seconds, with a standard deviation of 16.75 seconds. The paths of the 1,000 customers for each of the three layouts from this simulation took 10.6 hours to calculate. Improvements in processing speed could be achieved by using only the Euclidean distance. To create 1,000 customer paths and calculate distance using Euclidean distance only, it takes only 7 seconds, or on average .007 seconds per customer. The tradeoff is that Euclidean distance is a less accurate representation of the walking path. Another option is to constrain the graph search space by segmenting the image into overlaid nodes in an algorithmic way or in a crude way (like further downsampling the image). Finally, other path generation algorithms could be explored to increase efficiency.

In this section I showed how to compare the outputs of the simulation across different layouts using paired confidence intervals. For the example layouts, Layout 1 and 3 are both shown to generate similarly long walking paths. Using more samples, a narrower confidence interval could be obtained potentially differentiating one from the other. The Euclidean distance was shown to have different results than the A* path distance, which indicates it is not always a good representation for walking distance. Finally, a timing analysis was completed showing the speed of computing the A* path is fairly slow for the 500x500 pixel images, and options for future exploration on improving efficiency were presented.

Conclusions

This project applied simulation and computer vision techniques to provide a data-driven method for evaluating early-stage store layout plans. I showed how to parse a store diagram into a computerized representation, generate simulated customers, compute their walking path and distance, and compare the results for two example layouts. This analysis was repeated on real proposed layouts from a local co-op, and results were provided to the Store Development Committee.

There are many avenues for future work on this project. One area where this simulation could be improved is through tailoring the shopping list creation to the expected behavior of customers. Customer preferences are known to vary by age, region, and other factors, and the co-op planning committee expects behavior to vary from that shown in larger grocery chains. As more is known about prospective customers, this simulation can be changed to incorporate those changes and create more realistic scenarios. As the store moves forward in inventory planning, a more detailed store layout could be used, and other shelf allocation techniques applied. This information again increases accuracy, and also opens up the possibility of moving from walking path lengths into specific information about the products customers pass. Combined with data on impulse buying this could translate into an estimate of the monetary increase in sales. Finally, an important next step in any model that's based primarily on approximations and assumptions is to collect real data and compare the true results to the projected results. In this case collecting data won't be possible in the near term, or anytime for all but one of the layouts, but it provides potential for future analysis, such as that done in [3, 10].

In conclusion, in early phases of store planning simulation can provide the opportunity to make data-driven decisions even when data is scarce. In this project, I showed how to simulate customer walking paths to compare which proposed store layout is likely to generate the highest amount of impulse buying.

Works Cited

- [1] Bhadury, Joyendu, et al. "Store Layout Using Location Modelling To Increase Purchases." www.acsu.buffalo.edu/~batta/batta%20et%20al.pdf.
- [2] Campbell, Jeff. "What Is the Profit Margin for Grocery Stores?" *The Grocery Store Guy*, 30 Apr. 2020, thegrocerystoreguy.com/what-is-the-profit-margin-for-grocery-stores/.
- [3] Dorismond, Jessica P. "Data-Driven Models for Promoting Impulse Items in Supermarkets." *University at Buffalo, The State University of New York*, 2019.
- [4] Dorismond, Jessica. "Supermarket Optimization: Simulation Modeling and Analysis of a Grocery Store Layout." *2016 Winter Simulation Conference (WSC)*, 2016, doi:10.1109/wsc.2016.7822385.
- [5] Hart, Peter, et al. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, 1968, pp. 100–107., doi:10.1109/tssc.1968.300136.
- [6] Hoch, Stephen J., and George F. Loewenstein. "Time-Inconsistent Preferences and Consumer Self-Control." *Journal of Consumer Research*, vol. 17, no. 4, 1991, p. 492., doi:10.1086/208573.
- [7] Hui, Sam K, et al. "The Traveling Salesman Goes Shopping: The Systematic Deviations of Grocery Paths from TSP Optimality." *Marketing Science*, vol. 28, no. 3, May 2009, pp. 566–572., doi:<https://doi.org/10.1287/mksc.1080.0402>.
- [8] Kollat, David T., and Ronald P. Willett. "Customer Impulse Purchasing Behavior." *Journal of Marketing Research*, vol. 4, no. 1, 1967, pp. 21–31., doi:10.1177/002224376700400102.
- [9] Larson, Jeffrey S., et al. "An Exploratory Look at Supermarket Shopping Paths." *International Journal of Research in Marketing*, vol. 22, no. 4, 2005, pp. 395–414., doi:10.1016/j.ijresmar.2005.09.005.
- [10] Ozgormus, Elif, and Alice E. Smith. "A Data-Driven Approach to Grocery Store Block Layout." *Computers & Industrial Engineering*, vol. 139, 2020, p. 105562., doi:10.1016/j.cie.2018.12.009.
- [11] Psparks. "Instacart Market Basket Analysis." *Kaggle*, 20 Nov. 2017, www.kaggle.com/psparks/instacart-market-basket-analysis.
- [12] "Psychology of the Grocery Store: USC Online." *USC MAPP Online*, 12 Mar. 2020, appliedpsychologydegree.usc.edu/blog/psychology-of-the-grocery-store/.
- [13] "The Instacart Online Grocery Shopping Dataset 2017." *Instacart*, www.instacart.com/datasets/grocery-shopping-2017.
- [14] "What Is a Co-Op?" *Neighboring Food Co-Op Association*, nfca.coop/definition/.