Contemporary Engineering Sciences, Vol. 16, 2023, no. 1, 71 - 79
HIKARI Ltd, www.m-hikari.com
https://doi.org/10.12988/ces.2023.93121

Computational Techniques for Locating Industrial Products in Warehouses

Sotirios Tsakiridis

Department of Computer, Informatics and Telecommunications Engineering,
International Hellenic University - Serres Campus,
62124, Serres, Greece

Apostolos Papakonstantinou

Department of Civil Engineering and Geomatics
Cyprus University of Technology, Lemesos, Cyprus

Alexandros Kapandelis

Department of Computer, Informatics and Telecommunications Engineering
International Hellenic University - Serres Campus
62124, Serres, Greece

Paris Mastorocostas

Dept. Informatics and Computer Engineering
University of West Attica, Egaleo, Greece

Alkiviadis Tsimpiris

Department of Informatics, Computer and Telecomunications Engineering
International Hellenic University,
62124, Serres, Greece

Dimitrios Varsamis

Department of Computer, Informatics and Telecommunications Engineering,
International Hellenic University - Serres Campus,
62124, Serres, Greece

Abstract

The computational estimation of an indoor or open-space warehouse inventory is based on the prior knowledge of the pallet dimensions. The input data consist of a three-dimensional point-cloud created by three-dimensional (3D) scanners (LiDAR technology) adapted to aerial vehicles (Drones). For research purposes, a storage simulator has been implemented in the Python language (version 3.9). In the first phase, this research focuses on the ideal case of point-dispersion, with integer values for coordinates, in which a unit length corresponds to the distance between two neighboring pixels in the horizontal or vertical direction. In a subsequent stage, the generator of three-dimensional points will be modified to produce more realistic warehouse models. Improved versions of existing algorithms will be proposed, taking into consideration the height variations.

Keywords: Point cloud, 3D scanners, Warehouse modeling

1 Introduction

The integration of automation technologies in various inventory warehouses is considered essential for global industry, to ensure smooth and effective operation and organization of storage warehouses. The significance of integrating automation technologies is particularly pronounced, especially nowadays, due to the crucial impact of globalization. Numerous global industrial companies, with a wide range of products, require live control and monitoring of their distribution warehouses in the most modern and beneficial way. This can be achieved by leveraging technology to its fullest extent.

Automated data creation and knowledge extraction from product repositories has been of extensive interest to the scientific community in recent years. The rapid development of aerial vehicles (drones) combined with their relatively low cost of acquisition and operation, has significantly contributed to this direction. In the work of [1], the different types of aerial vehicles are described and categorised according to their size, weight and energy consumption, to make the right choice in each use case.

In parer [2] the authors successfully describe the simultaneous control of several autonomous navigation robots in an industrial warehouse environment.

In paper[3] the authors propose a method for the estimation of the volume of a material that is randomly scattered in warehouses.

The work of [4] use a GNSS device mounted on an unmanned aerial vehicle to estimate the volume of four piles of wood chips using image processing algorithms.

The specific approaches initially require data acquisition and subsequently involve processing the data to derive the results. In this study, we focus on simplifying the algorithmic methods without compromising the accuracy of the results. The aim is to reduce the computational complexity, making an algorithm suitable for real-time application. Data collection is performed using LIDAR (LIght Detection and Ranging) sensors adapted to unmanned aerial vehicles (UAVs).

In this paper, the methodology of data processing is presented at a theoretical level. Issues related to sensors and drone technical specifications, as well as the determination of optimal sampling frequencies and flight speeds, will be addressed in future research.

2 Methodology: Brute Force

The method of brute force [5] thoroughly examines all points in space. A necessary condition is the arrangement, according to geographical longitude and latitude. Consequently, the points in space are reduced to a kind of two-dimensional grid, in which the rows number of a cell corresponds to the x component of the point, whereas the columns number corresponds to the y component and the value of the cell to the z component. In simulation conditions, the values of the components are integers.

Let the $P = \hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n$ as a finite set of three-dimensional vectors, where each vector corresponds to a point in the cloud with $\hat{p}_i = (x_i, y_i, z_i)$. Initially, two ordered sets $\mathbb{X}$, $\mathbb{Y}$ are respectively created for the discrete values of the components x and y. Their creation occurs in two steps. In the first step, the unique values of x and y for all points in the cloud are recorded in two sets and in the next step these sets are sorted in ascending order. The classification of the components in the two sets is done by a single pass of the cloud points, meaning it has a complexity of $O(n)$. For the classification of the sets, the "merge sort" algorithm has been chosen with a time complexity of $O(nlogn)$. Thus, the overall complexity of the ordered sets for the components x, y created in this phase is $O(nlogn)$.

A hash table is created to rank the value of the z component of the cloud points. An encoding of the x, y components of the point in an alphanumeric (string) of the form "(x, y)" is used as a key. The hash function is defined as follows:

$$f : "(x_i, y_i)" \rightarrow z_i, \quad \forall \hat{p}_i = (x_i, y_i, z_i) \in P$$

The time complexity of hash table creation is $O(n)$, while the recovery com-

plexity in the general case is $O(1)$.

The ordered sets $\mathbb{X}$, $\mathbb{Y}$, along with the array representing the fragmentation of the z component, create the desired arrangement of points in the cloud on a grid, meaning, a Cartesian coordinate system is formed with axes representing the values of the sets and $\mathbb{Y}$, where at position (x_i, y_i) and $(x_i + w, y_i + l)$ exists a height value z. The algorithm for calculating the number of pallets in a point cloud is following

```
1. procedure COUNT_PALLETS(P, w, l, h)
2.   X  []
3.   Y  []
4.   HASH  {}
5.   PALLETS  0
6.   for p in P do
7.     if p.x not in X then
8.       X.append(p.x)
9.     end if
10.     if p.y not in Y then
11.       Y.append(p.y)
12.     end if
13.     key  (p.x, p.y)
14.     HASH[key]  p.z
15.   end for
16.   merge_sort(X)
17.   merge_sort(Y)
18.   for i1 to LENGTH(X) do
19.     for j1 to LENGTH(Y) do
20.       EXISTS, HEIGHT  CHECK_STACK(i, j, X, Y, HASH, w, l)
21.       if EXISTS then
22.         PALLETS  PALLETS + HEIGHT/h
23.         REMOVE_STACK(i, j, X, Y, HASH, w, l)
24.        end if
25.     end for
26.   end for
27.   return PALLETS
28. end procedure
```

The stack check algorithm at position (i, j) (`CHECK_STACK`) is presented below.

```
1. procedure CHECK_STACK(i, j, X, Y, HASH, w, l, dx, dy)
2.     (X[i],Y[j])
3.   z  HASH[KEY]
```

```
4.   if z = NULL or z = 0 then
5.     return FALSE, 0
6.   end if
7.   k  i + 1
8.   width  0
9.   while X[k] - X[i] <= w do
10.       (X[k],Y[j])
11.     if HASH[KEY] = z then
12.       width  X[k] - X[i]
13.       k  k + 1
14.     else
15.       break
16.     end if
17.   end while
18.   k  j + 1
19.   length  0
20.   while [k] - Y[j] <= l do
21.       (X[i],Y[k])
22.     if HASH[KEY] = z then
23.       length  Y[k] - Y[j]
24.       k  k + 1
25.     else
26.       break
27.     end if
28.   end while
29.   if width >= w  2*dx and length >= l  2dy then
30.     return TRUE, z
31.   else
32.     return FALSE, 0
33.   end if
34. end procedure
```

Since the scanning of space by LIDAR sensors is done incrementally with a specific and predetermined frequency, it is unlikely to detect points exactly at the boundaries of the stack.

Assuming dx is the minimum distance between two consecutive width scans, which depends on the maximum sampling frequency of the LIDAR and the flight speed of the UAV. If the expected width of the stack is $\mathbb{W}$, then the worst-case scenario leading to the maximum deviation is to detect a point, before or immediately after the start and end of the stack, respectively. Thus, the maximum deviation that can be observed is $2dx$. The same applies to lengthwise scanning. These parameters are considered in the `CHECK_STACK` algorithm to detect stacks with widths and lengths slightly smaller than expected based

on the scanning frequency.

We should also consider the occasion where neighboring stacks exist, as shown indicatively in Figure 1.

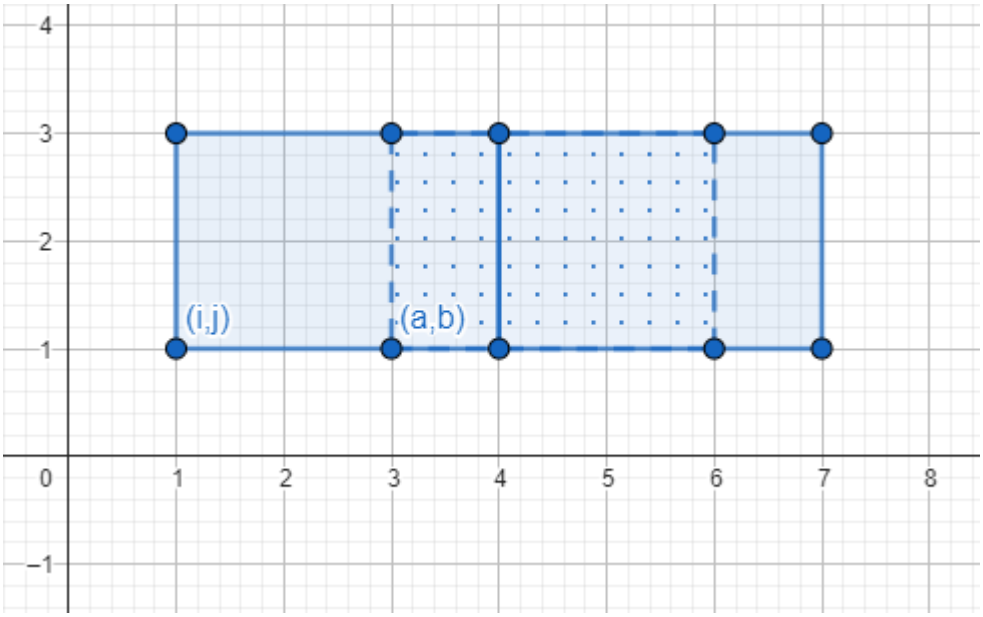

Figure 1: The detection of two neighboring stacks

If the current check point is (a, b), it is evident that a non-existent stack will be detected, due to the points overlapping issue occurred from the two neighboring stacks. To address this case and given that the algorithm will examine the point (i, j) before (a, b), due to the arrangement of and $\mathbb{Y}$ sets, in case a stack is detected at position (i, j), it will be removed from the hash table by removing all keys of points belonging to it, except for the points on the top and right side, which may belong to the beginning of a neighboring stack. Thus, the point (a, b), will not be checked again. Figure 2 represents three adjacent stacks (A, B, and C) and depicts the points that have been removed from the hash table after detecting stack A.

The stack removal algorithm (`REMOVE_STACK`) is following

```
1. procedure REMOVE_STACK (i, j, X, Y, HASH, w, l)
2.   m  i
3.   n  j
4.   while X[m] - X[i] < w do
5.     while Y[n] + Y[j] < l do
6.         (X[m],Y[n])
7.       HASH.remove(KEY)
8.       n  n + 1
```

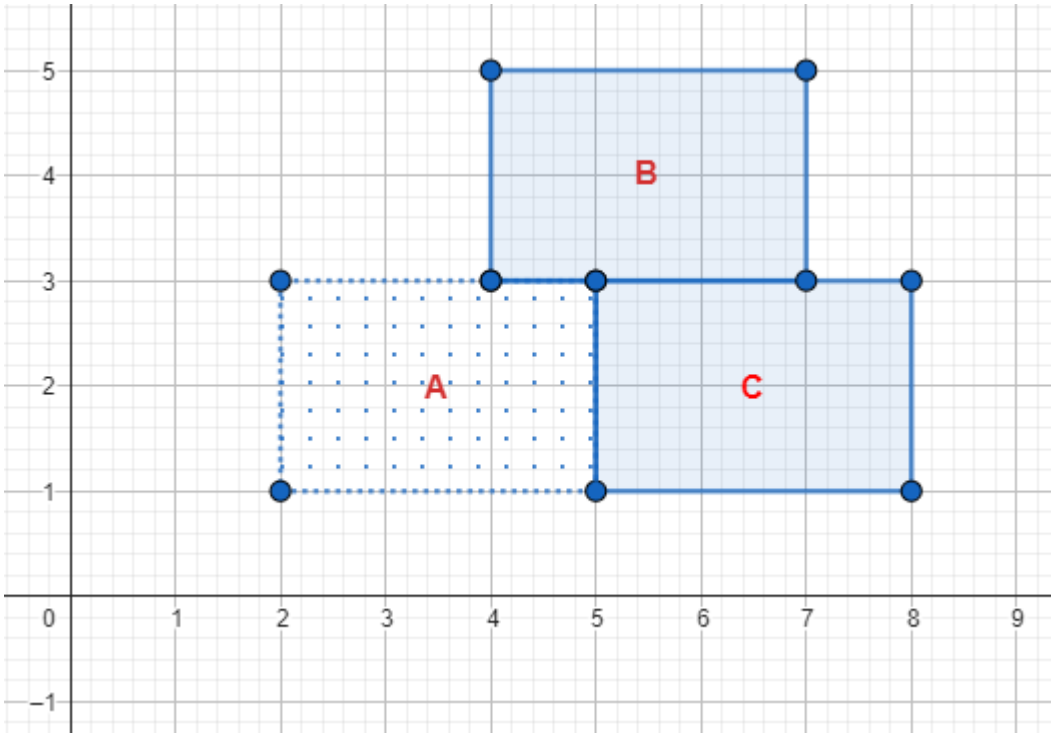

Figure 2: The detection of three neighboring stacks

```
9.      end while
10.      m  m + 1
11.      n  j
12.    end while
13. end procedure
```

3 Time complexity study

The minimum time complexity of the counting algorithm is obviously (n), since each of the n points in the point cloud must be examined at least once. The two while loops in the `CHECK_STACK` algorithm, theoretically, raise the time complexity to $O(n^2)$. However, practically, this does not occur. The two loops only check the two sides of the stack and not the intermediate points, and the dimensions of the stack are much smaller compared to the warehouse. Thus, with a predetermined and constant stack size, the `CHECK_STACK` algorithm has practically constant complexity $O(1)$. A similar situation applies to the `REMOVE_STACK` algorithm. The two nested while loops in lines 4 and 5 can theoretically be executed n times, making the overall complexity of the algorithm $O(n^2)$. The extreme theoretical scenario occurs in the case of a

stack size equal to that of the warehouse, which obviously does not hold. The much smaller dimensions of the stack compared to the warehouse and the fact that `REMOVE_STACK` is executed only in the case of stack detection make its time complexity practically constant, i.e., (1). Therefore, we can practically consider that the time complexity of the counting algorithm is (n) which, combined with the sorting complexity of the point cloud, results in an overall time complexity of the proposed methodology as $O(nlogn)$.

4 Conclusion Future Research

The requirement for parallel alignment of stacks with the scanning direction is quite restrictive. In the case of disorderly placement of stacks in the warehouse, the proposed methodology fails. If the current examined point has a height greater than the reference point (meaning it does not belong to the warehouse floor), we cannot consider it as the bottom-right corner of a potential stack. The while loops of the `CHECK_STACK` algorithm need to be modified to detect the orientation of the sides. One obvious approach could be the recursive examination of neighbors of the point under consideration to find the stack's orientation. However, this dramatically increases the time complexity, making the algorithm practically infeasible. Therefore, a different approach needs to be followed.

Following the research, algorithms will be studied and evaluated for converting point clouds into images with brightness values representing the z-component. This enables the application of image processing techniques from the scientific field of digital image processing to detect coherent regions that might belong to stack positions.

To apply the algorithm in real-world conditions, small fluctuations in the values of point components should be considered. In the ideal case examined, the component values were integers. In practice, small deviations in these values are expected for points that need to be categorized at the same height, width, or length. The warehouse simulation algorithm will be appropriately modified by introducing error parameters for the three components. Components of points with values within the error bounds will be considered equal.

Finally, parallelization techniques will be tested to reduce execution times and enable real-time application. The specific implementations of the `CHECK_STACK` and `REMOVE_STACK` algorithms are suitable for parallel execution since there are no linear dependencies. Ideally, parallelization could occur on a graphics card (GPU). Modern GPUs have hundreds of cores capable of executing basic calculations in parallel. However, the primary goal of the work is to apply the algorithms "on-the-fly." Data collection and processing will be done by microcontrollers or microcomputers (e.g., Raspberry Pi) adapted for aerial vehicles. These microcomputers inherently have limited computational power

and graphics cards with few cores. Parallelization will be done at the CPU level. Obviously, in a second stage, processing of the overall data can be performed "off-the-fly" on computing systems that offer parallelization capabilities on the GPU.

Acknowledgements. This research work was carried out as part of the project "Optimization of placement and counting products in large industrial areas using UAV" (Project code: 6-0083129) under the framework of the Action "Investment Plans of Innovation" of the Operational Program "Central Macedonia 2014 2020", that is co-funded by the European Regional Development Fund and Greece.

References

[1] B. Vergouw, H. Nagel, G. Bondt, B. Custers, Drone technology: types, payloads, applications, frequency spectrum issues and future developments, *The Future of Drone Use,* New York, NY: Springer, 2016, 21-45. https://doi.org/10.1007/978-94-6265-132-6_2

[2] HR. Everett, DW. Gage, GA. Gilbreath, RT. Laird, RP. Smurlo, Real-world issues in warehouse navigation, Wolfe WJ, Chun WH, eds. Mobile Robots IX, Vol 2352, Bellingham,Washington: International Society for Optics and Photonics, *SPIE Proceedings*, 1995, 249-259. https://doi.org/10.1117/12.198975

[3] C. Arango, CA. Morales, Comparison between multicopter uav and total station for estimating stockpile volumes, *ISPRS Int Arch Photogramm Remote Sens Spatial Inf Sci.*, **1** (2015), 131-135. https://doi.org/10.5194/isprsarchives-xl-1-w4-131-2015

[4] M. Mokros, M. Tabacak, M. Lieskovsky, M. Fabrika, Unmanned aerial vehicle use for wood chips pile volume estimation, *Int Arch Photogramm Remote Sens Spat Inf Sci.*, **41** (2016), 953. https://doi.org/10.5194/isprs-archives-xli-b1-953-2016

[5] Marijn Jh Heule, Oliver Kullmann, The science of brute force, *Communications of the ACM,* , **60** (2017), no. 8, 70-79. https://doi.org/10.1145/3107239

Received: December 7, 2023; Published: December 23, 2023