AI-assisted synchronization of maps from 2D to 3D: method and application

Mac Van Vien*

Institute of Information Technology, Academy of Military Science and Technology.
*Corresponding author: vienmvit@gmail.com
Received 12 Oct. 2023; Revised 10 Dec. 2023; Accepted 12 Dec. 2023; Published 30 Dec. 2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.CSCE7.2023.37-49

ABSTRACT

In this article, we introduce a method for synchronizing operations from 2D digital mapping software to 3D digital sand table software using AI technology. This method solves the challenges and difficulties of combining 2D and 3D in military applications, such as the difference between 2D and 3D space, the accuracy and efficiency of the synchronization process, and the user experience when working with three-dimensional space. This method also uses AI models to detect and classify objects on the terrain and estimate the Z coordinates of objects on 3D sand tables. This method has been applied in the T3BD system, including 2D (GISDesktop) and 3D (GlobeDeskop) applications, and has been used by several units in the military. Experimental results show that this method achieves low synchronization times for all types of objects, from simple to complex. This method can improve the accuracy and efficiency of the synchronization process and enhance the user experience when working with three-dimensional space.

Keywords: Map; Digital Map; 2D; 3D; Sand table; 3D digital sand table; Synchronization.

1. INTRODUCTION

Spatial information is very important for military operations, as it allows for clear and accurate representation of the terrain, the enemy, and the friendly forces. Currently, there are two types of software to display information in military operations: 2D digital maps (2D) and 3D digital sand tables (3D). Each type of software has its own advantages and disadvantages. 2D can represent military symbols and scenes clearly and accurately, but it cannot simulate three-dimensional space. 3D can display three-dimensional spatial information vividly and intuitively, but it is difficult to represent military symbols and scenes in 3D. Therefore, combining 3D with 2D is a useful solution to enhance the presentation and interaction with three-dimensional space [1]. However, there is a significant difference between spatial information in 2D and 3D. 2D only provides the X and Y coordinates of the objects on the map and has a fixed view towards the north, while 3D also provides the Z coordinate of the objects and allows for different viewing angles, such as yaw, pitch, and roll. Both types of software are graphics-intensive and require high performance, so they need to be organized to avoid overload.

The two types of software usually run independently from each other and communicate with each other through a real-time synchronization module. If one software has an error, it does not affect the other software. Based on these contents, the main goal of this paper is to propose a new method to synchronize operations from 2D to 3D using artificial intelligence (AI) techniques. Our method consists of the following main components:

- The overall diagram of the method of synchronizing operations from 2D to 3D presents the general information flow of the solution.
- Data structure: presenting the data structure of the objects to be synchronized and

converting from that data structure to a message format to transmit messages.

- The process of receiving messages from 2D and classifying messages.
- The process of executing messages in 3D is the most important component; it uses AI to solve the challenges of the difference between 2D and 3D without losing information or the performance of 3D.

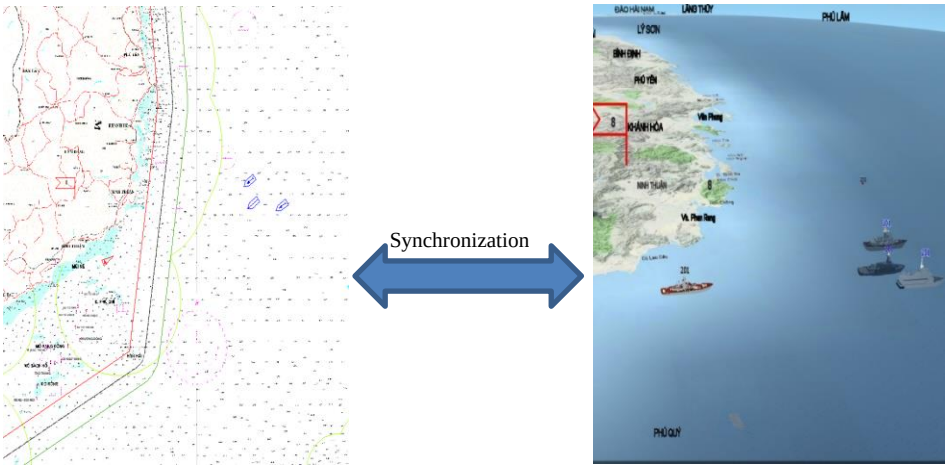

Figure 1. *Synchronously draw symbols from 2D to 3D.*

2. PROPOSED METHOD

This paper proposes a new method to synchronize operations from 2D to 3D based on a real-time synchronous communication module. The method consists of the following steps:

- Designing a data structure for the objects to be synchronized and converting it to a message format to transmit messages.
- Transmit real-time operational information from 2D to 3D.
- Receive synchronously from 2D and classify messages by the type of action, such as drawing, moving, rotating, deleting, etc.
- Perform synchronously in 3D, using artificial intelligence to identify the location of the action, solving the Z coordinate problem in 3D.

The synchronization process is shown in figure 2. The process is divided into two independent processes: synchronization receiving and synchronization executing, which use a concurrent queue to store and exchange synchronization data. This process solves the problem of operational synchronization between 2D and 3D because:

- When users operate in 2D, the operation information is transmitted to 3D through the real-time synchronization module [4], which is responsible for connecting and exchanging information between the two software. This module ensures that if one of the two software programs has an error, it will not affect the other software.
- The concurrent queue is a data structure that allows data to be added, deleted, and updated simultaneously without errors. It is used to store the operation information received from 2D and provide it to 3D for execution.
- The synchronization receiving process is performed when there is information from the synchronization module, packaging the received information and putting it into the queue. This packaged data is designed to avoid duplicate information and to be easy

to search for and update.

- The synchronization execution process is performed in the main loop of the 3D application, thus ensuring real-time synchronization execution. At each main loop of the 3D application, it takes data from the queue, checks if there is enough information to execute, if not, returns the queue to receive more information, and if yes, then executes. We use artificial intelligence to identify the location of the action, thereby solving the Z coordinate problem in 3D.

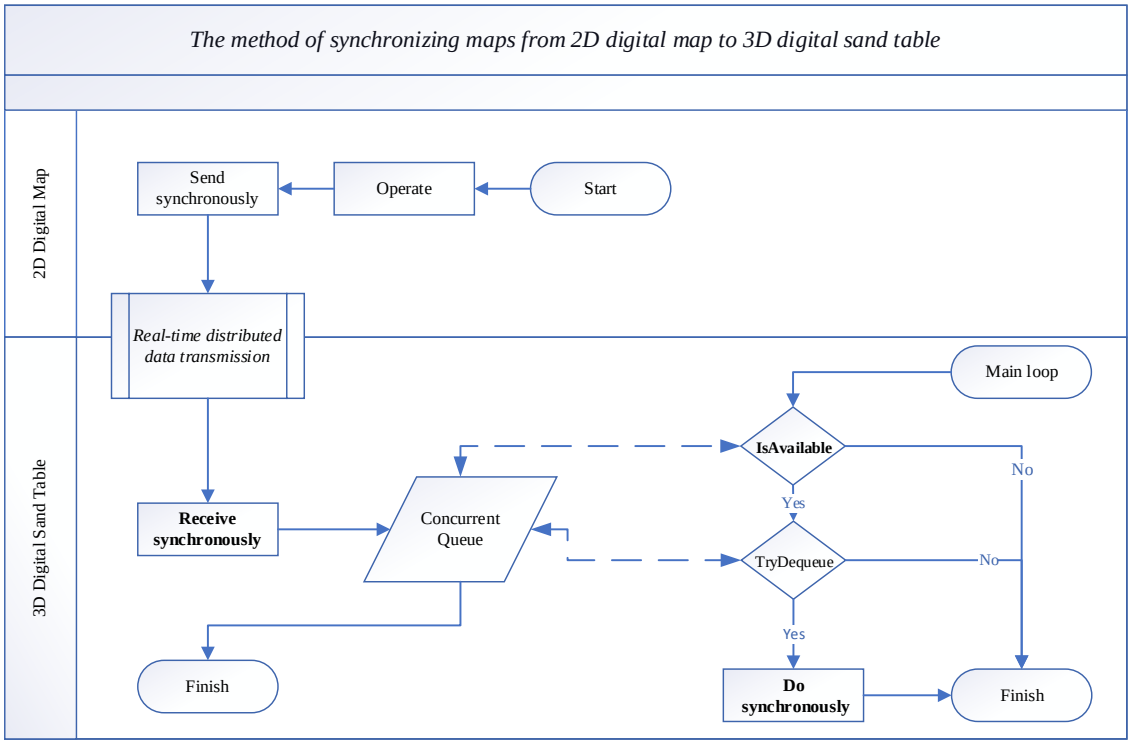

Figure 2. *The synchronization processes.*

2.1. Data Structure

To synchronize operations between a 2D and a 3D, both software programs need to understand each other's data structures. The data structure represents and stores the spatial information of symbol objects, scenes, objects in scenes, and 3D software controls. A common data structure is needed to avoid errors, losses, or distortions of information during synchronization.

2.1.1. Data Structure of Military Symbols

In the 3D digital sand table [3], a symbol is an object representing a combat object, a unit, an action, an event, or a state in three-dimensional space. A symbol includes the following information: Symbol ID is the unique value in the sand table. Library symbol ID is the value to identify the symbol type in the list of symbol types, for example, tank, rocket, rocket ship, etc. Symbol type is the type of symbol shape; Position information is a three-dimensional vector (x,y,z) to determine the position of the symbol on the sand table. Altitude is an attribute to indicate whether the symbol is measured by altitude above the ground, above sea level, or following the ground surface. Direction is a three-

dimensional vector (yaw, pitch, roll) to determine the direction of the symbol on the sand table; Objects are components for drawing symbols on the sand table, which can be 3D models, 2D shapes, 3D shapes, or combinations of them; Content labels are components to represent unit version, object name, quantity, state, etc.

A complex symbol can include many objects and multiple content labels. A symbol can also contain other symbols as its subcomponents. For example, a tank battalion can be represented by a symbol containing three individual tank symbols.

The design of synchronous data for the digital sand table must ensure that the information about the symbol and its subcomponents is updated and transmitted accurately and quickly. Therefore, the data structure of the symbol is organized in a tree structure [2], as shown below:

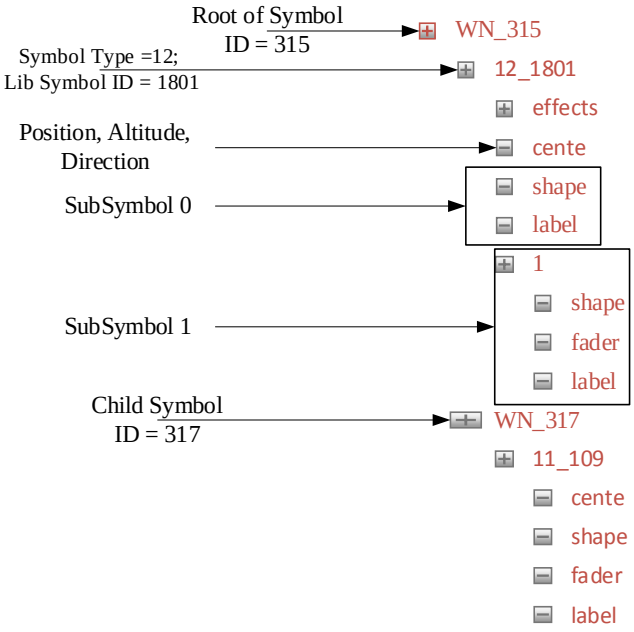

Figure 3. *Example of Symbol data.*

Figure 3 shows an example of a complex symbol. The data structure for this symbol is: symbol ID = 315; symbol type =12 (directionless); library symbol ID = 1801 (tank platoon); Position information = cente's position; altitude = cente's altitude; direction = cente's direction; effects Subsymbol 0 (shape, label), Subsymbol 1 (shape, fader, label), Child Symbol (ID = 317, Symbol type =11 (point), Library symbol ID = 109 (headquarters of tank platoon), Position information = cente's position, Altitude = cente's altitude, Direction = cente's direction, Subsymbl 0 (shape, lable, fader)).

2.1.2. Data Structure of Slides and Objects

A slide is a concept used to refer to the content that needs to be presented in 3D digital sand table software. Slides can help users create and present scenarios or events in a clear and engaging way. Slides can also help users focus on the key points or messages that they want to convey. A slide can include multiple objects that are symbols with added actions or effects, such as hide/show, motion, 3D effects, etc. A slide also has

attributes such as a descriptive name, time, illustration content, illustration sound, illustration image, and illustration audio.

We define a slide as a list of data items that includes: slide ID, parent slide ID, name, camera position, begin action, end action, slide duration, date time, weather, slide media, slide notes, slide images, show layers, and slide's objects. A slide's object as a list of data items that includes: Object ID, slide ID; symbol ID, type, moving path, begin state, end state, effects.

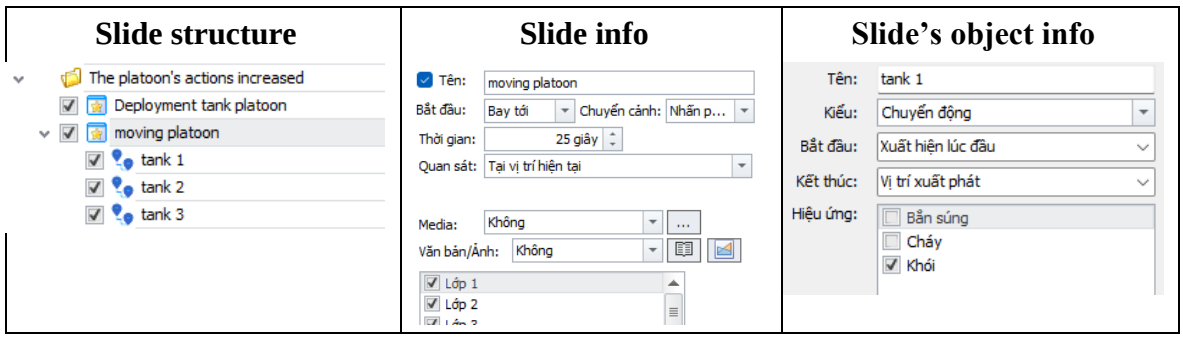

Figure 4. *Example of Slide and Slide's Object.*

Figure 4 shows an example of a slide that describes the action of a tank platoon attacking in 3D digital sand table software. In this slide example, there are three objects (tank symbols) on 3D digital sand table software. They have motion action and smoke effect added to them. The data structure for this example is: Slide info (Name= "moving platoon", Begin Acion = "Fly to", End Acion = "Key press", Duration = 25, Camera Position = "Current Position", Media = None, Notes/Images = None, Show Layers = All); Slide's Obect info (Name = "tank 1", Type = "Moving object", Begin State = "Appear", End State = "Begin Position", Effects = "Smoke").

2.1.3. Data structure for communication

Our method uses a real-time communication module and a mapping table to ensure the consistency and efficiency of the synchronization process. Between 2D and 3D software, there is a module for real-time communication. This module allows each operation's information from 2D to be packaged into a message sent to 3D, and the 3D software to analyze the message and execute it.

Based on the data structure of objects, synchronization messages are divided into three types: symbol synchronization, scene and object synchronization, and control synchronization. The first field of the message is MsgName, which defines the message type, and MapAction, which defines the operation type (add, edit, delete, etc.).

We utilize a mapping table to streamline the synchronization process between 2D and 3D, preventing unnecessary repetition of object synchronization messages. For instance, a single 2D symbol can be synchronously sent to 3D multiple times. Similarly, when a user aims to create an object within a slide corresponding to a symbol identified as x in 2D, the system retrieves the symbol x in 3D and maps it to its corresponding symbol y. By storing this mapping information, redundant messages are eliminated, ensuring consistency between symbols across both interfaces.

Table 1. *Message structrure.*

Message Type		Message structure	Description
Symbol	Symbol	*MsgName|MapAction|MapLayerID|MapParentSymbolID|LibSymbolID|SubSymbolIDs|MapSymbolID|ObjInfoID*	MsgName = SynLibSymbol, MapLayerID – Layer's ID that contains the symbol on 2D. SubSymbolIDs – number of subsymbols that symbol has
	Sub symbol	*MsgName|MapAction|MapLayerID|MapStyleID|MapParentSymbolID|SymbolType|MapSymbolID|Position|Direction|Caption|GeomPointCount|GeomPointX|GeomPointY|SubMapSymbolID|Style*	MsgName = SynSubSymbol, SubMapSymbolID – subsymbols's ID, GeomPointCount|GeomPointX|GeomPointY - Gemetry information for shape objects, Caption - label content.
Slide		*MsgName|MapAction|ParentMapAniSlideID|MapAniSlideID|SlideName|SlideType|SlideOrder|BeginAction|EndAction|X|Y*	MsgName = SynAniSlide, |X|Y - coordinates on the map of the viewport center.
Slide's Object		*MsgName|MapAction|MapLayerID|MapAniSlideID|MapAniObjectID|Name|MapSymbolID|AniType|BeginState|EndState|BeginTime|Duration|AniPathX|AniPathY*	MsgName = SynAniObject, AniPathX|AniPathY – The coordinates of the motion path of the object in 2D.
Controls		*MsgName|ControlInfo*	MsgName = SynMapExtent, SynAniSelection,... ControlInfo – data to execute control commands.

Table 2. *Mapping table structrure.*

Mapping table	Field name	Description
SynSymbolMapping	GlobeSymbolID	Symbol's ID on 3D
	MapSymbolID	Symbol's ID on 2D
	MapParentSymbolID	Parent Symbol's ID on 2D
SynSlideMapping	GlobeSlideID	Slide's ID on 3D
	MapSlideID	Slide's ID on 2D
SynAniObjectMapping	GlobeAniObectID	ID of slide's object on 3D
	MapAniObjectID	ID of slide's object on 2D

2.2. Operation 1: Receive synchronously

For a SymbolCommand, it will be executed when there is complete information, including the general information about the slide and the information from the SubSymbols, i.e., a complete SymbolCommand will be concatenated with the number SubSymbolIDs (see item 2.1.3) of SubSymbolComand. So, we use two lists: one is the PendingSymbol list (which stores the symbol commands that do not have enough subcommands); the other is the PendingSubSymbol list (which stores the subcommands that have not been matched to any symbol commands). The synchronous data receiving algorithm is described below.

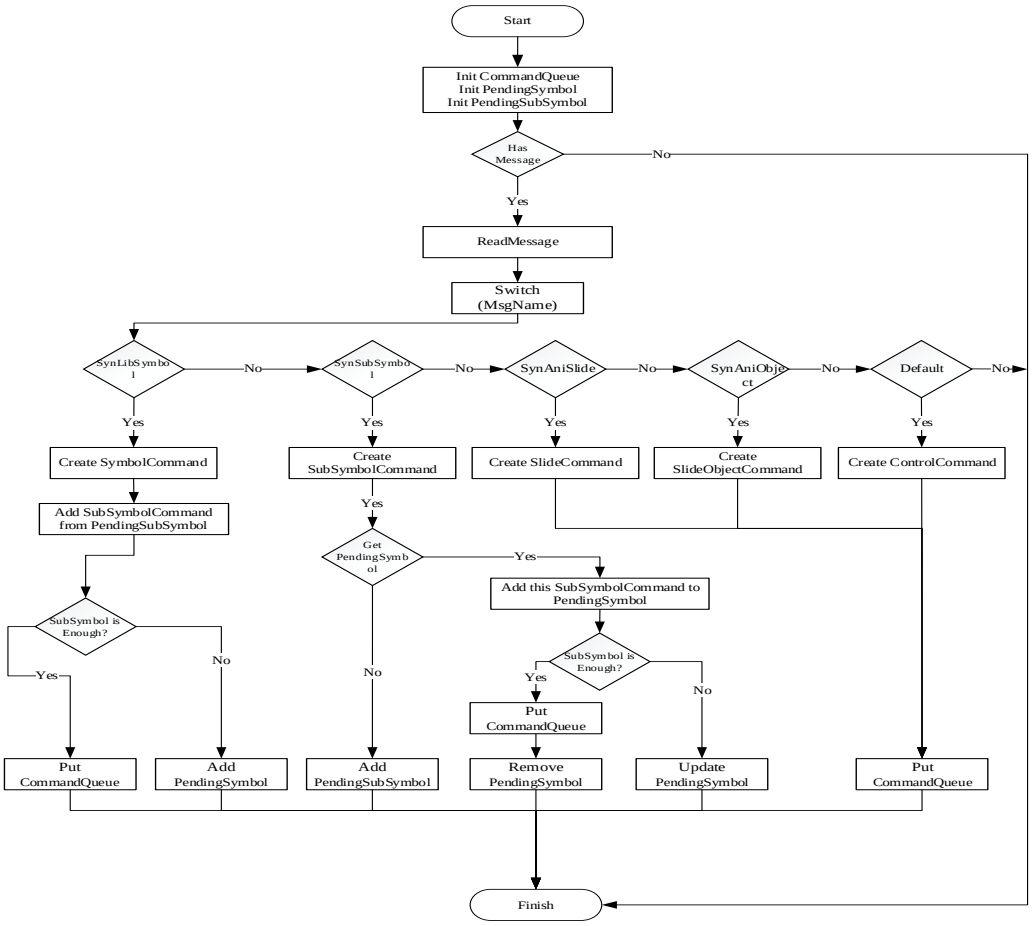

Figure 5. *The algorithm for receiving synchronously.*

2.3. Operation 2: Perform synchronously

We apply the Command Pattern [5] in building command objects to execute messages. Objects in a queue consist of objects of the SymbolCommand, SlideCommand, SlideObjectCommand, and ControlCommand. All of them implement the "Execute()" method to execute the operation and the "TravelCount" property to count the number of passes.

At each main loop of the 3D application (the On_Frame loop), it always checks the conditions to execute the commands. The method name is "IsAvailable()", and its purpose is to ensure the consistency and efficiency of the synchronization process. These conditions include: There is a command waiting in the loop; Currently, no command is being executed; The software's performance is guaranteed through the current main loop parameter (in the application, the time of a main loop < 1/30 sec is satisfied).

2.3.1. Execute command

It takes each command in the queue in turn to execute command to execute the operation. **Receive Synchronously** and **Do Synchronously** are parallel. Therefore, we must check if the parent object exists or not based on the Mapping table data (Sec. 2.1.3). If there is no parent object, it returns to the queue and increments the TravelCount variable until it reaches a certain number of times but still cannot find the parent object.

Then it removes the command from the queue. The schematic diagram of the algorithm is shown in figure 6.

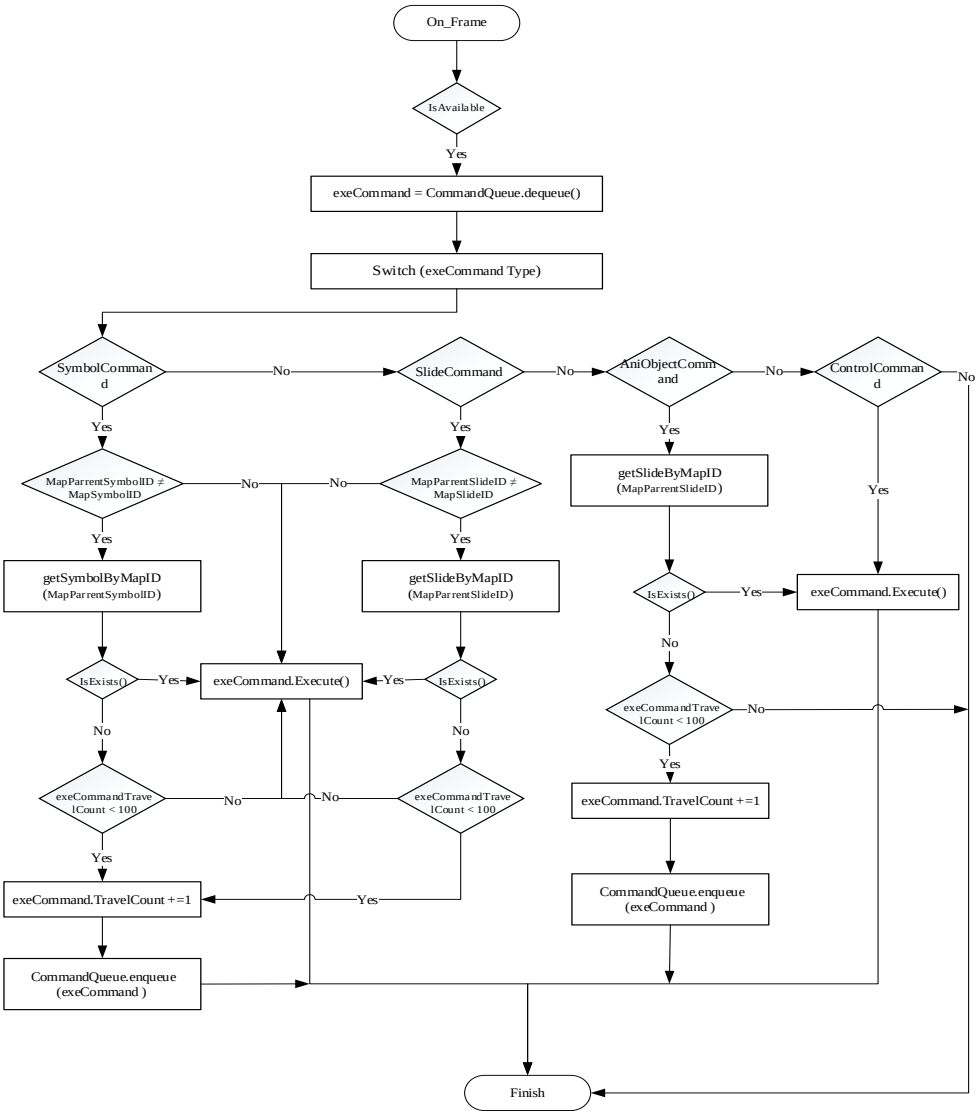

Figure 6. *The algorithm for execution synchronously.*

2.3.2. Height Estimation of the Objects using AI tools

In this section, we describe our proposed method for synchronizing the operations from 2D software to 3D software using AI techniques. Our method consists of two main steps: object detection and classification, and height estimation. We use AI models to detect and classify the terrain objects that the operational objects are located on. We then use different methods to determine the 3D coordinates of each object based on its 2D coordinates, height, and height datum. The height datum is the reference level for measuring the height of the objects, such as on terrain, terrain relative, or terrain absolute. We use different methods to convert the 2D coordinates to 3D coordinates, using the height values as the Z coordinate for each type of object. Figure 7 shows an overview of our method.

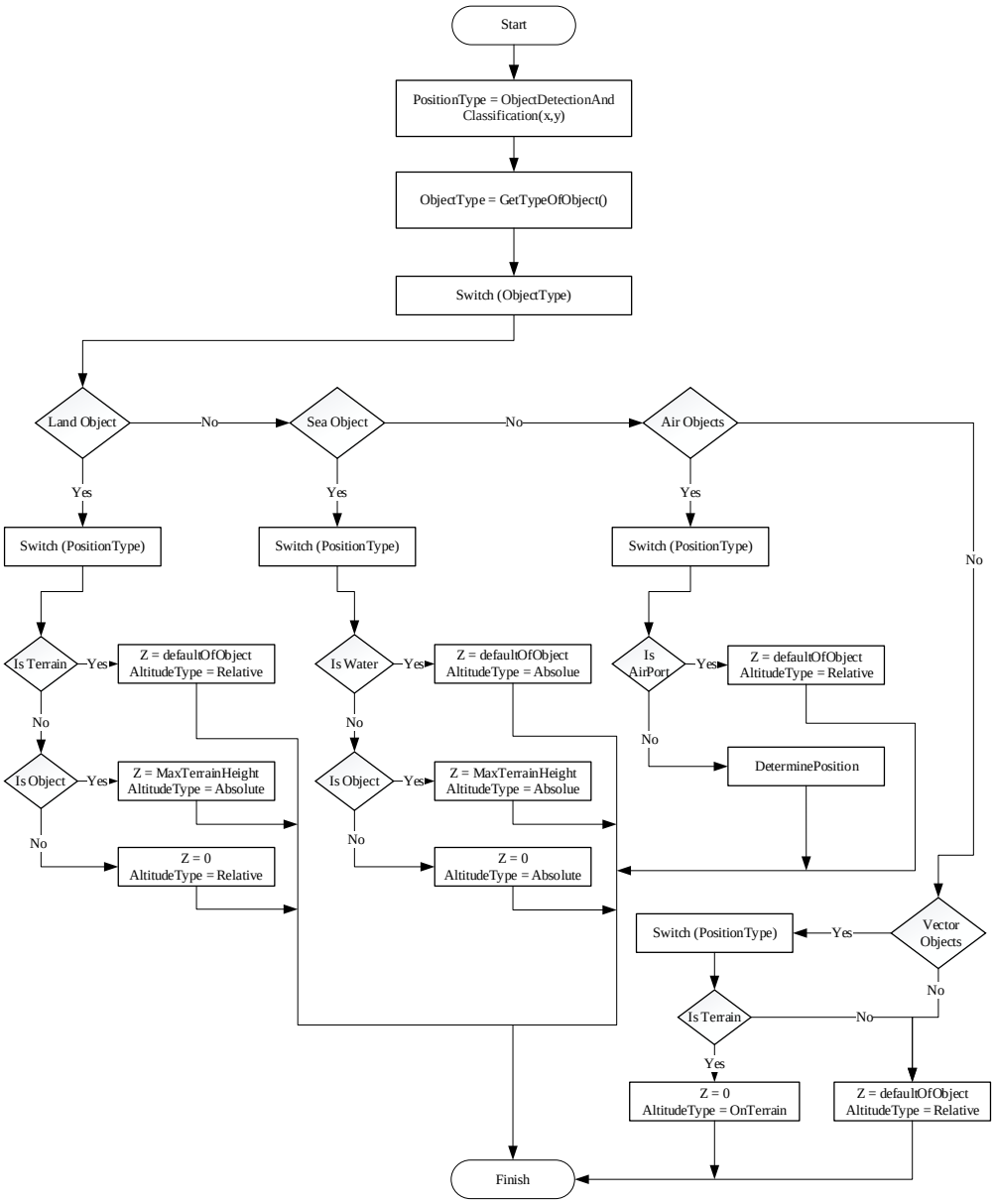

Figure 7. *The method of using AI to Estimate the Z Coordinate.*

Object detection and classification: The first step of our method is to identify and classify the objects on the satellite image map, such as vegetation, sea, airport, port, ships, planes, vehicles, etc. We use an AI model based on the YOLOv4 algorithm [6], which is a state-of-the-art object detection algorithm that can process images in real time, and implement with ML.NET [7]. The AI model takes the satellite image map as input and outputs the label and location of each object on the image, as well as a confidence score that indicates how confident the model is about the detection and classification.

Height estimation: The second step of our method is to estimate the height and height datum of each object based on its label and location on the satellite image map. The height datum is the reference level for measuring the height of the objects, such as on terrain, terrain relative, or terrain absolute. Different types of objects have different

height datums, depending on their position and movement. For example, land objects usually have a terrain relative height datum, which means their height is measured from the ground level. Sea objects and air objects usually have a terrain absolute height datum, which means their height is measured from the sea level. We use different methods to determine the height and datum of different types of objects, such as land objects, sea objects, air objects, vector objects, and other types. For each type of object, we use the information from the satellite image map, the object detection and classification model, and the DeteminePosition method [8] to estimate the height and height datum of the object. Figure 8 shows an illustration of the output of the method for determining the height and height datum of a warship and a tank.

Figure 8. *An illustration of the output of the method for determining the height and height datum.*

3. APPLICATION AND EXPERIMENTS

3.1. Implementation

Our method is applied for the T3BD system [1], including 2D applications (GISDesktop) and 3D applications (GlobeDeskop) (see figure 1). When deploying, two applications are either installed on two computers connected over a LAN or on the same computer with two monitors. The system has been used by a number of units in the military. It is evaluated to meet the professional service requirements of the units with high speed.

The system allows users to synchronize objects between 2D and 3D applications in real time. Users mainly only need to work in 2D for the work they usually do, while the content of the operation is synchronized to 3D. This way, users can use 2D to get an overview of the objects on a map and 3D to get a detailed and realistic view of the objects in a scene. The system also provides a 3D presentation function that enables users to create a report that includes both 2D and 3D views of the objects. This function makes the report more intuitive and informative for the users and their audiences.

3.2. Experimental Criteria, Environment, and Scenario

Experimental Criteria: An important criterion for method evaluation is synchronization time. Synchronization time is the time that elapses from the end of the operation in 2D to the end of the execution in 3D. The user request response time corresponding to each object must satisfy:

- Simple notation operation time (only subsymbol 0): ≤ 0.3 seconds;
- Complex symbol operation time: ≤ 0.5 seconds;

- Control operation time ≤ 0.1 seconds;
- Slide operation time ≤ 0.1 seconds;
- Time to work on the slide's object ≤ 0.2 seconds.

Experimental Environment: We use the T3BD system and tools to measure the synchronization time. The specific environment is described in the table 3:

Table 3. *Experimental Environment.*

Software	**Map/Terrain**	**Computer info**	
		OS	**Hardware info**
GlobeDesktop	*Province: Khanh hoa*	*Windows 10 Pro N, version 21H2, 64 bit.*	*Processor:Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz . Ram: 16.0 GB*
GisDesktop	*Province: Khanh hoa Ratio: 1: 50 000*	*Windows 10 Pro N, version 21H2, 64 bit.*	*Processor:Intel(R) Core(TM) i7-8750H CPU @ 2.20GH.Ram: 16.0 GB*
T3BD Communication	*Version 1.0*		

Experimental scenario: We perform tasks in 2D software synchronously to 3D and measure the working time for the following cases:

- Case 1: Operate one simple symbol 10 times in a row;
- Case 2: Send synchronously 10, 100, and 200 simple symbols simultaneously;
- Case 3: Operate one complex symbol 10 times in a row;
- Case 4: Send synchronously 10, 100, and 200 different complex symbols simultaneously;
- Case 5: Operate one operation to control the sand table 100 times in a row to calculate the average time;
- Case 6: Operate one slide 10 times in a row;
- Case 7: Operate one slide's object 10 times in a row;
- Case 8: Send synchronously 10, 20, and 50 slides, with each slide having one object;
- Case 9: Send synchronously 10, 20, and 50 slides, with each slide having five objects.

3.3. Experimental results, and discussion

Table 4. *Test results for Experimental.*

Case	**Object type**	**Number of objects**			**Synchronization Time (seconds)**			**Eligibility (seconds)**		
1	Simple symbol	1			0.25			≤ 0.3		
2	Simple symbol	10	100	200	2.61	25.76	44.25	≤ 3	≤ 30	≤ 60
3	Complex symbol	1			0.39			≤ 0.5		
4	Complex symbol	10	100	200	3.8	32.85	55.07	≤ 5	≤ 50	≤ 100
5	3D control	1			0.07			≤ 0.1		
6	Slide	1			0.08			≤ 0.1		
7	Slide's object	1			0.12			≤ 0.2		
8	Slide with one object	10	20	50	2.01	3.57	7.96	≤ 3	≤ 6	≤ 15
9	Slide with five objects	10	20	50	5.5	7.39	15.53	≤ 11	≤ 22	≤ 55

The test results confirm exceptional synchronization times across all object types:

- Simple Symbol maintained synchronization within 0.25 seconds for a single symbol, escalating proportionally as the quantity increased. Remarkably, synchronizing 200 symbols achieved within 44.25 seconds (0.22 seconds/symbol).
- Complex symbols showcased synchronization times well below the threshold, meeting the criteria even with 200 symbols within 55.07 seconds.
- 3D control commands consistently operated within 0.1 seconds, attesting to the system's efficiency in this domain.
- Slide synchronization, including objects within them, consistently met predefined criteria.

The obtained results affirm several critical aspects of the proposed method: Accuracy and Correctness: All operations executed from 2D were accurately reflected in the 3D environment, confirming the correctness and reliability of the synchronization method; Performance and Scalability: The method consistently met predefined criteria across various object types and quantities, demonstrating robust performance and scalability; Optimized Processing: Notably, the decrease in average operation time with increasing object quantities reflects the method's efficiency. The underlying mechanism within the 3D operation function efficiently identifies available symbols on the board, streamlining the copying process and reducing synchronization times.

The experimental outcomes strongly support the method's effectiveness in synchronizing objects between 2D and 3D environments across a spectrum of scenarios. The method not only meets stringent synchronization time criteria but also demonstrates adaptability and scalability, laying a strong foundation for its practical application in military contexts.

4. CONCLUSIONS

This paper proposes a method to synchronize art from a 2D digital map to a 3D digital sand table. Our method solves the challenges and difficulties of combining 2D and 3D in military applications, such as the difference between 2D and 3D space, the accuracy and efficiency of the synchronization process, and the user experience when working with three-dimensional space. The article proposes a common process consisting of two interdependent processes: receiving synchronization and executing synchronization. Then present the data structure design content to the main algorithms of the method in section 3.1. One of the main features of our method is the use of AI techniques to overcome the differences between 2D and 3D space, especially in estimating the Z coordinate of the objects on the 3D map. By using AI, our method can improve the accuracy and efficiency of the synchronization process, and enhance the user experience when working with three-dimensional space.

This method has been applied in the T3BD system [1], including 2D application (GISDesktop) and 3D application (GlobeDeskop). The system has been used in a number of units in the military and has received positive feedback from the users. The test results of the method show that it achieves low synchronization times for all object types, from simple to complex, as shown in table 4. The proposed method has proven to

be effective and feasible when applied to a number of units in the army.

In the next phase, the method needs to be improved and enhanced in the following directions: Continue to build editing algorithms to speed up synchronous execution, intelligentize the synchronous execution function by using contextual information to reduce the distinction between 2D and 3D space, and apply artificial intelligence techniques such as speech recognition for synchronous operation and control.

REFERENCES

[1]. Nguyen Duc Dinh, Hoang Van Toan. "*System Design Documentation of T3BD System*", (2020).
[2]. "*TerraExplorer Programmer Guide – Basic concepts*", Skyline Sofware Inc, (2017).
[3]. Nguyen Canh Hung, Nguyen Duc Dinh, Le Ngoc Tu "*Application of the skyline technology platform in the construction of 3d digital sand table data for command and consultation work in sea and island warfare*", Academy of Military Science and Technology, (2019).
[4]. Le Van Diep, Hoang Van Toan. "*Real-time distributed data transmission and application in data synchronization of sea and island battle plans based on digital charts and 3d digital sand tables"*, Academy of Military Science and Technology, (2019).
[5]. w3sDesign.com. *"The Command design pattern - Problem, Solution, and Applicability"*. (2017).
[6]. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. *"YOLOv4: Optimal Speed and Accuracy of Object Detection"*, (2020).
[7]. Nikola M. Zivkovic. "*Machine Learning with ML.NET – Object detection with YOLO*", (2021).
[8]. Mac Van Vien. *"The method to determine position in 3d digital sand table"*, (2022).

TÓM TẮT

Đồng bộ tác nghiệp từ bản đồ số 2D sang sa bàn số 3D: phương pháp và ứng dụng sử dụng trí tuệ nhân tạo

Trong bài viết này, chúng tôi giới thiệu phương pháp đồng bộ hóa các thao tác từ phần mềm bản đồ số 2D sang phần mềm sa bàn số 3D sử dụng công nghệ AI. Phương pháp này giải quyết những thách thức và khó khăn khi kết hợp 2D và 3D trong các ứng dụng quân sự, chẳng hạn như sự khác biệt giữa không gian 2D và 3D, độ chính xác và hiệu quả của quá trình đồng bộ hóa cũng như trải nghiệm của người dùng khi làm việc với không gian ba chiều. Phương pháp này còn sử dụng mô hình AI để phát hiện, phân loại vật thể trên địa hình và ước tính tọa độ Z của đối tượng trên sa bàn số 3D. Phương pháp này đã được áp dụng trong hệ thống T3BD, bao gồm các ứng dụng 2D (GISDesktop) và 3D (GlobeDeskop), và đã được một số đơn vị trong quân đội sử dụng. Kết quả thử nghiệm cho thấy phương pháp này đạt được thời gian đồng bộ thấp cho mọi loại đối tượng, từ đơn giản đến phức tạp. Phương pháp này có thể cải thiện độ chính xác và hiệu quả của quá trình đồng bộ hóa và nâng cao trải nghiệm người dùng khi làm việc với không gian ba chiều.

Từ khóa: Bản đồ; Bản đồ số; 3D; Sa bàn; Sa bàn số 3D; Đồng bộ tác nghiệp.