Human-robot collaborative assembling task planning by using digital twin

Paula Luque-Contreras
Dpto. de Ing. de Sistemas y Automática
Universidad de Málaga
Málaga, Spain
0009-0008-4024-0582

Álvaro Galán-Cuenca
Dpto. de Ing. de Sistemas y Automática
Universidad de Málaga
Málaga, Spain
0000-0002-2644-6460

Juan María Herrera-López
Dpto. de Ing. de Sistemas y Automática
Universidad de Málaga
Málaga, Spain
0000-0003-3692-3088

Marcos Rollón
Dpto. de Ing. de Sistemas y Automática
Universidad de Málaga
Málaga, Spain
0009-0004-8877-252X

Víctor F. Muñoz
Dpto. de Ing. de Sistemas y Automática
Universidad de Málaga
Málaga, Spain
0000-0003-2404-0050

***Abstract*—This paper explores human-robot collaboration in assembly tasks using a digital twin (DT) and virtual reality (VR) technology to enhance efficiency and safety. A task planning algorithm based on A* is introduced to determine the optimal sequence of assembly operations. The system incorporates a UR3e robot and colored prisms in an experimental setup, simulating collaborative tasks. The methodology includes task scheduling and assignment within the DT, with results evaluated using Visual Components Premium. This approach demonstrates improvements in task allocation, highlighting the significance of precision and collaboration in shared workspaces.**

***Index Terms*—human-robot collaboration, digital twin, virtual reality, task planning, A* algorithm.**

I. Introduction

Assembly tasks involve placing parts in specific locations and are characterised by the handling of many components, frequent interruptions, and short cycles. However, with the rise of mass customisation and globalisation, manual methods alone can no longer keep up [1]. Market demands have driven automation forward, as robots offer the ability to work without fatigue, with repeatability, precision, and high speed. Nevertheless, although robotic assembly is more efficient and less costly, the lack of flexibility prevents it from performing some specific activities [2].

As a result, human-robot collaboration has emerged as a solution for these types of tasks. Advances in sensor technology and communication protocols have given rise to a new generation of robots, known as cobots. These kind of robot is lighter, more flexible, and designed for safer, more efficient integration into shared workspaces facilitating the collaboration [3].

To improve this collaboration, in addition to using cobots, it is essential to implement software capable of determining the optimal sequence of actions needed to achieve a given goal in the shortest possible time (task planner).

This work was supported by the Spanish Ministry of Science and Innovation under Grant TED2021-129893B-I00.

An important step after this planning is task assignment, which will depend on the available resources, time (both activity and rest), task complexity, coherence, which refers to whether the assembly direction of adjacent parts is the same during the assembly process as well as the characteristics of the objects to be assembled [3], [4].

A technology related to task planning and task allocation is Digital Twin (DT). They are defined as a virtual representation of a physical system, its associated environment and processes, which are updated through the exchange of information between the physical and virtual systems [5], [6]. From the existing literature, it can be inferred that DT helps reduce costs while improves efficiency, safety, and human confidence [7]. Furthermore, it serves to encapsulate algorithms, provides support for decision-making and robot control, creates planning algorithms [8] and functions as a predictor or a fault control [9]. Because of all these reasons, DTs are used to train and to check the correct functioning of the task planning, as well as to study the efficiency in implementing task allocation [10].

Currently, there are several platforms available for creating and using digital twins (DT), such as Siemens MindSphere, Microsoft Azure Digital Twins, and GE Digital Twin for Predix. These platforms offer comprehensive solutions that include IoT connectivity, real-time data analysis, simulation, and visualisation. In contrast, Visual Components Premium specialises in 3D simulation and modelling of manufacturing processes. It provides advanced tools for robot programming and validation, along with an extensive library of industrial components. For this work, which focuses on human-robot collaboration in assembly operations, Visual Components Premium was chosen for its specialised approach, accurate simulation of collaborative interactions, and its support for using various communication protocols.

This paper shows how to create a task planning system based on the A* Algorithm. It has focused on collaborative assembling tasks between a human and a robot, and how to

integrate and test its functionality within a DT developed in Visual Components Premium. The description of these processes is provided in the following sections: Section II outlines the experimental environment and the structure of the task planner and task assignment. Section III details the creation of the task planner. Section IV presents the implementation process, while Section V describes the conducted experiments. Finally, Section VI draws conclusions.

II. Approach to a Collaborative Task

This work focuses on creating a task planner in a DT for human-robot collaboration. Aiming to simulate assembling process, the task involves coloured prisms stacked on a table, which must be placed in goal positions by stacking them on top of another prism or placing them on the work table.

To achieve this, the experimental environment shown in Figure 1 has been created. Label A shows a close-up of blocks (prisms of different colors) and places (dark grey rectangles) where blocks can be positioned. To identify places, they have been named with the letter 'p' and numbered from left to right; for blocks the letter 'b' has been used and numbered from left to right and from bottom to top.

In addition, it appears in label B a UR3e collaborative robot with a suction cup as end effector, in label C a screen displaying instructions for the operator and the robot's actions, in label D a tool used to manipulate the pieces when interacting with the DT through virtual reality (VR) glasses and in label E a button on the table for the human operator to press upon completing their action.

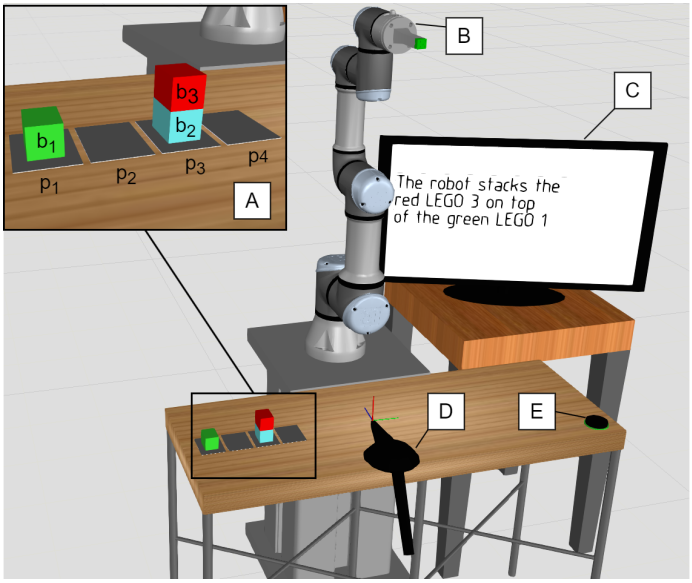

Fig. 1. Experimental setup.

To represent the environment, as it is oriented toward assembly tasks, the block world representation has been chosen. This consists of a simplified representation that uses two dimensions and includes two kinds of elements: blocks and places. Blocks are defined as objects that can be manipulated by the robot or by the human and change to another position, while places are possible positions where the blocks can be on the work surface.

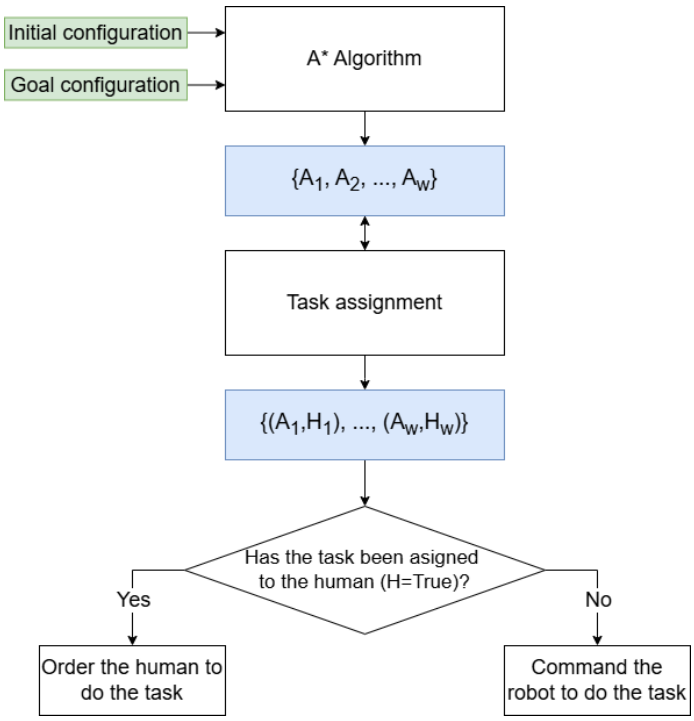

Fig. 2. Task planning and assignment flowchart

Also it is necessary to relate blocks and places. To define this relationship, it has been used the predicates: $Free(X)$ or $Above(X, Y)$. $Free$ indicates if X (that can be a block or a place), has a block on it or not, and $Above$ informed if Y (which is a block), is on top of X.

Once the setup has been designed and a way to represent it has been chosen, the next step is to create a search algorithm and assign actions to study the collaboration. An action is a task that change the actual configuration of the setup to a new one.

In this paper, the action chosen has been MOVE(X,Y,Z). It consists of moving the block X above the place or block Y on block or place Z. It can be applied if X and Z are free and X is on Y. After applying it, a new configuration of the setup will be created, it will meet the conditions X and Y free as well as X on top of Z.

Starting with an initial and a goal configuration, as shown in Figure 2, the A* algorithm is initiated. The algorithm begins by generating nodes, each representing a new configuration of the setup after moving a block. This is done by applying the MOVE(X,Y,Z) rule, which shifts any free block in the current configuration to a new position. Once A* completes its process, the resulting sequence of actions is returned as a vector: $\{A_1, A_2, ..., A_w\}$, where w represents the final action to be released.

After that, actions are assigned to robot or human, through a vector composed of the action along with who has

to do the that action. The nomenclature used has been $\{(A_1, H_1), ..., (A_w, H_w)\}$ where A is the action and H an indicator of who should do it, it will be True if the human releases the action or False if it will be done by the robot.

Finally, if H is True, the operator watch the instructions to follow in a screen in the DT through VR glasses, while if it is False, the robot is commanded to move or stack the block from a position to a new one.

III. Formalisation of the task for the A* algorithm

The goal of task planning is to complete tasks quickly and safely. It involves determining the order of sub-tasks (actions) from an initial configuration to a goal configuration.

Based on the block world, to represent the environment numerically, it is needed to begin by creating two vectors that indicate blocks and places that exist in the workspace. Blocks' vector appears in equation 1, where n is the total number of blocks present in the environment while places' vector in equation 2 where m is the total number of places.

$$B = (b_1, b_2, \ldots, b_n) \tag{1}$$

$$P = (p_1, p_2, \ldots, p_m) \tag{2}$$

To relate blocks and places, looking for to be able to represent the environment, it has been used the expression that appears in the equation 3, this is applied to x which can be a block or a place and returns the value i if the block b_i is above x or zero if x has no block above it (it is Free).

$$S(x) = \begin{cases} i & \text{if } b_i \text{ is above } x \\ 0 & \text{if } x \text{ is free} \end{cases} \tag{3}$$

From this expression, it arises the idea of representing each configuration j of the environment as a node N_j as it appears in equation 4. Each node j is composed of the pair of vectors $\widehat{B}_j$ and $\widehat{P}_j$. $\widehat{B}_j$ arises from applying the rule in equation 3 to each block in equation 1 and the same happens with the vector $\widehat{P}_j$ but by applying the rule to vector in equation 2.

$$N_j = \begin{cases} \widehat{B}_j = (S(b_1), S(b_2), ...S(b_n)) \\ \widehat{P}_j = (S(p_1), S(p_2), ...S(p_m)) \end{cases} \tag{4}$$

Once the representation of the environment has been established, it can be started the task planning, using A* algorithm. It combines uniform cost search with a heuristic that estimates the remaining cost to reach the goal. The evaluation function of A* is defined in the equation (5) as:

$$f(N_C) = g(N_C) + h(N_C) \tag{5}$$

where g(N_C) represents the accumulated cost from the initial node to the current node N_C, and h(N_C) is a heuristic estimate of the cost from the current node to the goal node.

To use this algorithm, it is needed to create three functions to calculate the cost, the heuristics and look for the neighbours node to start applying it.

In this paper, the cost has been calculated as the number of movements that should be made to get from the initial node to the actual node, it appears in equation 6, represented as k.

$$g(N_C) = k \tag{6}$$

While the heuristic has been defined as the minimum number of movements necessary to go from the current configuration to the final one. Its expression is shown in equation 7. It is calculated by comparing the difference between the vector of places for the current configuration ($\widehat{P}_C$) and the goal configuration ($\widehat{P}_G$), and the difference between the vector state of blocks for the current configuration ($\widehat{B}_C$) and the goal configuration ($\widehat{B}_G$).

This entire expression is multiplied by $\frac{1}{2}$ because moving a block causes two changes in the vectors: one at the position it leaves and another at the position it moves to. However, this is counted as a single movement.

There are two summations in the equation, one that goes from 1 to the total number of blocks (n), and another from 1 to the total number of places (m). These summations traverse the vectors to compute the differences.

And finally, it also appears the function δ (equation 8) which returns 1 if its two inputs ($input1$ and $input2$) are different, and 0 if they are the same.

$$h(N_C) = \frac{1}{2} * \left[\sum_{u=1}^{n} \delta(\widehat{B}_C, \widehat{B}_G) + \sum_{v=1}^{m} \delta(\widehat{P}_C, \widehat{P}_G)) \right] \tag{7}$$

where:

$$\delta(input1, input2) = \begin{cases} 1 & \text{if } input1 \neq input2 \\ 0 & \text{if } input1 = input2 \end{cases} \tag{8}$$

Going on with the third function: looking for neighbours node, it's necessary to start with the parent node (actual node). From here, there will be a neighbour node for every move that can be done from the actual configuration, that is, applying the action MOVE(X,Y,Z) to every block that is free.

Once the sequence of movements to be carried out has been calculated using A* algorithm, human and robot must be commanded to follow them. These instructions appear on the screen in the DT when using RV glasses so the operator knows what he has to do while he is aware of what the robot's next action will be.

There are two types of instructions constructed following the vector $\{(A_1, H_1), ..., (A_w, H_w)\}$, they will have the format: "Operator/Robot, move the green block 6 to place 4" or "Operator/Robot stacks the grey block 9 on top of the green block 6". This distinction is made so that the instructions to the operator are clear and so that he is aware of what the robot is going to do.

In addition, the instructions indicate the colour of the piece to make it easier for the operator to identify it and pieces have been assigned a number in case there are several of the same colour or the user is colourblind.

An example of using the A* algorithm implemented is presented below. First of all, it is necessary to define the initial and goal configuration using nodes (following equation 4). They appear in equation 9 for initial configuration and in equations 10 for goal configuration.

$$N_0 = \begin{cases} \widehat{B}_0 = (0,3,0) \\ \widehat{P}_0 = (1,0,2,0) \end{cases} \tag{9}$$

$$N_G = \begin{cases} \widehat{B}_G = (0,0,1) \\ \widehat{P}_G = (0,3,2,0) \end{cases} \tag{10}$$

The next step, it is A* algorithm, that starts from the initial node. It looks for the neighbour nodes by applying the rule MOVE(X,Y,Z) to the free blocks b1 and b3. The neighbours obtained appears in Figure 3, where the Actual Node has been represented in an oval as the Parent Node, and in rectangles are shown the configuration of every new neighbour after applying the rule that appears above them. The block that has been moved is highlighted in blue.

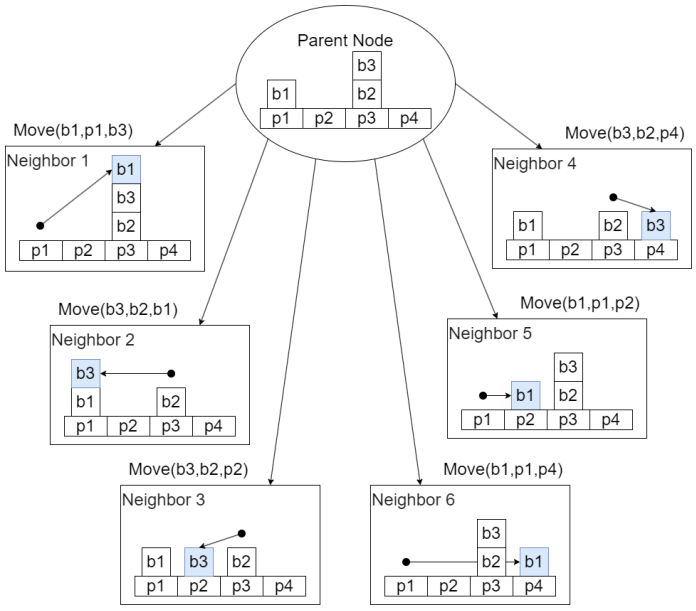

Fig. 3.

After that, new neighbours have been set out in the Table I, where the first column is the rule followed to arrive to the new configuration. The second one is the Node Name, this is to later distinguish which is the parent node. Third and fourth columns are the state of the blocks alongside the state of the places vectors to describe the position of every block. They follow the expressions mentioned in equation 4. Finally, the last column is to save the parent node.

Once the neighbours have been calculated, the cost and heuristic of every neighbour must be calculated too. The number of movements they have been done to go from the initial to the actual configuration is one because they are the initial configuration neighbours and the heuristic will be, following equation 7: 2 for neighbour 1, 4 for neighbour 2, 2 for neighbour 3, 4 for neighbour 4, 3 for neighbour 5 and 4 for neighbour 6. So the lowest cost *f*, calculated through equation 5, is 3 for the neighbours 1 and 3, so the algorithm will start exploring the both of them and will finally choose the less costly way.

TABLE I
PARENT NODE AND THEIR NEIGHBOURS.

Rule	Node Name	B_s	P_s	Parent Node
Parent Node	pn	(0,3,0)	(1,0,2,0)	0
Move(b1,p1,b3)	n1	(0,3,1)	(0,0,2,0)	pn
Move(b3,b2,b1)	n2	(3,0,0)	(1,0,2,0)	pn
Move(b3,b2,p2)	n3	(0,0,0)	(1,3,2,0)	pn
Move(b3,b2,p4)	n4	(0,0,0)	(1,0,2,3)	pn
Move(b1,p1,p2)	n5	(0,3,0)	(0,1,2,0)	pn
Move(b1,p1,p4)	n6	(0,3,0)	(0,1,2,0)	pn

With the cost, the heuristic and the neighbours node calculated, A* is applied until there is a solution or it returns empty if there is any. For this example, it finally returns the sequence: MOVE(b3,b2,p2), MOVE(b1,p1,b3).

IV. IMPLEMENTATION

To create the DT and integrate the A* algorithm in it, the sequence of steps shown in Figure 4 has been followed. It begins designing the task to be performed and the experimental setup. After that, the process can continue with Visual Components Premium program (everything carried out in this program has been highlighted in dark grey in the Figure). The first thing to do in this program is insert all components and if needed, connect signals.

With the entire setup positioned and connected, the next step is to use Visual Components Python Script, this is a kind of behaviour that works with specific commands of Visual Components in conjunction with Python 2.7 (highlighted in blue in the Figure).

Within the Python script, and looking for the integration of the A* algorithm in the DT, the vectors of blocks at initial and goal locations, and the vector of colours are introduced in this script as show in Figure 5.

It begins an initialisation function that creates the blocks, positions them based in the initial configuration, assigns them a mass to make it have physics and activates its 'VR Pickable' property so that it can be Interact with them in VR glasses.

The next step is to integrate A* into this Python script. To do that, they have been created 3 functions to calculate the cost, the heuristics and neighbours' node as it has been described in Section III. A* return a solution, which will be the order in which actions should be carried out if there is a solution ($\{A_1, A_2, ..., A_w\}$) or empty if it is not possible to arrive to the goal configuration, as mentioned in section II. Finally, if there is a solution the robot and the human are ordered to do their actions.

All this process allow to create the DT integrating the task planning in it. The next step is assign actions and use VR glasses to interact with the DT.

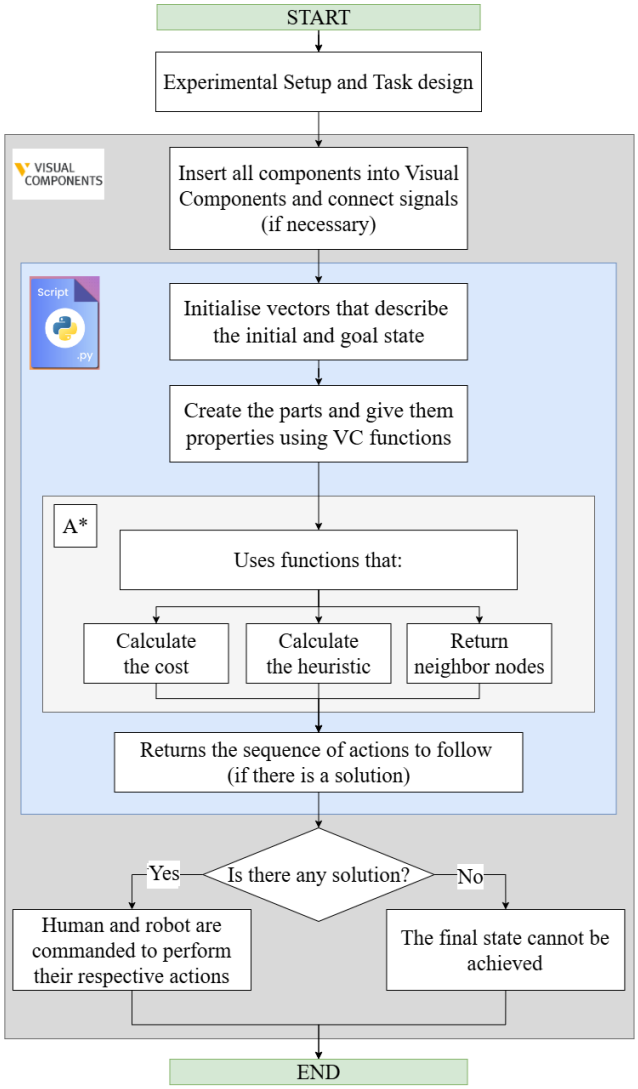

Fig. 4. Diagram with steps to follow to build the DT with the task planner integrated.

```
Initial_BlockState =  [0,3,0]
Initial_PlaceState = [1,0,2,0]
Colors = ["green", "cian", "red"]
Goal_BlockState =  [0,0,1]
Goal_BlockState = [0,3,2,0]
```

Fig. 5. How to introduce nodes for initial and goal configuration of the setup in Visual Components Python Script.

The glasses used at this paper have been Meta Quest 3. To connect them with the DT it has been used the programs: Visual Components Experience, SteamVR in the PC and Steam Link in the glasses.

V. Experiments

Last phase is experimentation to verify if A* algorithm works correctly in the DT, to interact with it using RV glasses, try two kind of action assignments and see the impact it has on the total time needed to complete the task.

These experiments consist of carrying out the actions from A*. There are two types of instructions: stack or place. 'Stack' is move a block on another block, while 'place' is move a block to an empty place on the table. The action of stacking requires greater precision because the block must be aligned with the block below it and the action of place is simply placing it on the table, in a larger space, so it does not need as much precision.

To assign the actions after A* has ordered them, it has been used two trials. **Trial 1** is human placing blocks and robot stacking them while **trial 2** is human stacking blocks and robot placing them.

These experiments have been tested in the environment which appears in Figure 1. The initial configuration appears in top of Figure 6 and the goal configuration in the lower part of the same Figure. These configurations of the setup have been chosen to include the actions: place a block to an empty place, stack a block on another which hasn't been placed or stack it on one which has already changed its position to test the difficulty of stacking a block on another which is non-parallel to others because the human has placed it to a place without the precision a robot has.

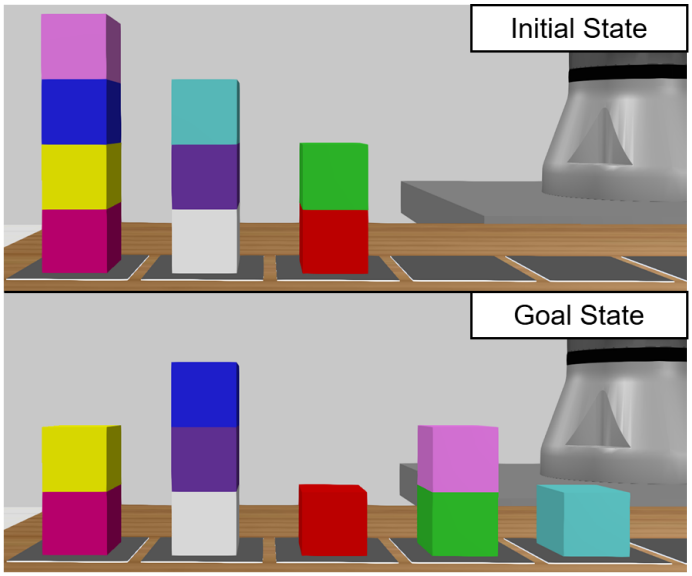

Fig. 6. Initial and Final configuration of the blocks.

Once the initial and goal configuration have been defined, A* algorithm calculates the order to follow to complete the task. It returns the sequence:

1) Human/Robot has to place green block 6 to place 4.
2) Human/Robot stacks pink block 9 on green block 6.
3) Human/Robot has to place cyan block 8 to place 5.
4) Human/Robot stacks grey block 7 on orange block 5.

And after the two trials to action assignment are tested, the time used to release these actions appears in Table II. In it, the first column is the actions that should be made, the second column is the methodology used and the third column is the time (in seconds) employed to do every action, calculated

doing an average of 10 experiments for every methodology by an only one subject.

TABLE II
ACTIONS FOLLOWING TWO TRIALS AND THE TIME TO RELEASE THEM.

Actions	Trial	Time (seconds)
Move green block 6 to place 4	1	9.2
	2	3.2
Stack pink block 9 on green block 6	1	3.2
	2	16.6
Move light blue block 8 to place 5	1	11.8
	2	3.2
Stack grey block 7 on orange block 5	1	2.8
	2	16

These experiments have not been developed on a physical setup yet, but they have been tested on VR glasses connected to the DT running on Visual Components Premium. How it is to release these actions using VR is shown in Figure 7 where part 1, 2 as well as 3 show the human placing the block to a place and pressing the button using the tool in Visual Components. Parts 4 and 5 the robot collects a block, stacks it on another, while in part 6 a "Task completed" message appears when the task has been finished.

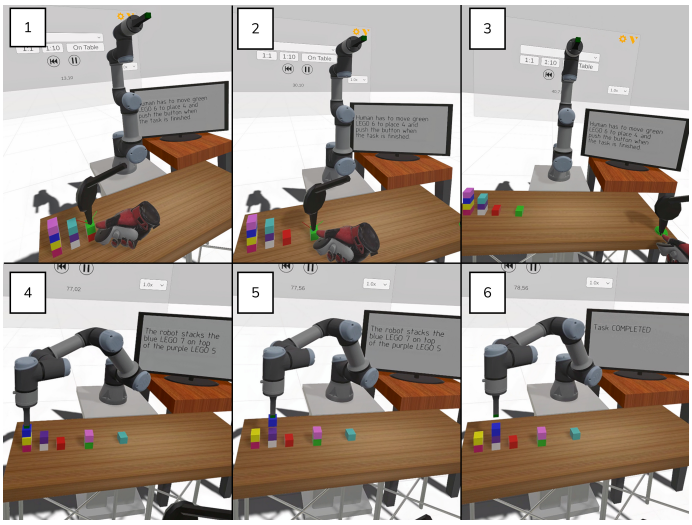

Fig. 7. Part of the tasks performed during the experiment in which the human moves the pieces and the robot stacks them.

VI. CONCLUSION

In this work, an A* task planning has been developed in a digital twin within the Visual Components Premium program. After this, in order to test its operation, an experimentation phase was carried out in which VR glasses are used to interact with the DT. The glasses allow safe, remote training without interrupting production. Operators can train repeatedly to prepare for real tasks. However, they face limitations, such as using controllers instead of hands, which requires additional training before interacting with the DT.

Of the training phase it can be conclude that after testing two types of task assignment, the time was 27 seconds when the human places the blocks and the robot stacks them compared to a time of 39 seconds when the human stacks blocks and the robot places them. This demonstrates the importance of task assignment and highlights the robot's effectiveness in precision tasks.

Future work aims to test different task assignments to find the fastest way to perform the task, connect the DT to the real experimental setup and environmental reconnaissance to bring the physical configuration to the DT.

Additionally, the heuristic and cost will be modified. The goal is to add more scoring systems that take into account factors such as part drops and mistakes placing a part in the wrong location to improve the deployment of the whole task.

REFERENCES

[1] A. Malik and A. Bilberg, "Complexity-based task allocation in human-robot collaborative assembly," *Industrial Robot*, vol. 46, no. 4, pp. 471–480, 2019.

[2] M. Raessa, J. C. Y. Chen, W. Wan, and K. Harada, "Human-in-the-loop robotic manipulation planning for collaborative assembly," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1800–1813, 2020.

[3] Y. Wang, J. Wang, J. Feng, and et al., "Integrated task sequence planning and assignment for human–robot collaborative assembly station," *Flex Serv Manuf J*, vol. 35, pp. 979–1006, 2023. [Online]. Available: https://doi.org/10.1007/s10696-022-09479-2

[4] A. Nkulu and M. Bahubalendruni, "Human-robot collaborative task planning for assembly system productivity enhancement," *Robotic Intelligence and Automation*, vol. 44, no. 1, pp. 120–130, 2024.

[5] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision Support Systems*, vol. 145, p. 113524, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923621000348

[6] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Digital twin paradigm: A systematic literature review," *Computers in Industry*, vol. 130, p. 103469, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361521000762

[7] E. VanDerHorn and S. Mahadevan, "Digital twin: Generalization, characterization and implementation," *Decision Support Systems*, vol. 145, p. 113524, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923621000348

[8] S. Liu, X. V. Wang, and L. Wang, "Digital twin-enabled advance execution for human-robot collaborative assembly," *CIRP Annals*, vol. 71, no. 1, pp. 25–28, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000785062200018X

[9] H. Li, W. Ma, H. Wang, G. Liu, X. Wen, Y. Zhang, M. Yang, G. Luo, G. Xie, and C. Sun, "A framework and method for human-robot cooperative safe control based on digital twin," *Advanced Engineering Informatics*, vol. 53, p. 101701, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034622001604

[10] P. Zhao, J. Liu, X. Jing, M. Tang, S. Sheng, H. Zhou, and X. Liu, "The modeling and using strategy for the digital twin in process planning," *IEEE Access*, vol. 8, pp. 41 229–41 245, 2020.