PUZZLING ENGINE: A DIGITAL PLATFORM TO AID THE REASSEMBLING OF FRACTURED FRAGMENTS

R.D.L. Hernandez,* S. Vincke, M. Bassier, L. Mattheuwsen, J. Derdeale and M. Vergauwen

Dept. of Civil Engineering, TC Construction - Geomatics
KU Leuven - Faculty of Engineering Technology
Ghent, Belgium
(roberto.delimahernandez,jens.derdaele,maarten.bassier,stan.vincke,lukas.mattheuwsen,maarten.vergauwen)@kuleuven.be

Commission II, WG II/8

KEY WORDS: Digital reassembling, Serious games, Heritage fragments segmentation

ABSTRACT:

The reassembling of fractured fragments is a paramount task in the fields of digital heritage documentation and reconstruction of archaeological artifacts and monuments. This process is typically carried out by manually puzzling matching clues such as decoration,shape, contour, etc. This labor poses a challenge for restorers as fractured fragments are fragile, deteriorated and in some cases bulky. In order to aid experts in this meticulous and time-consuming process, a puzzling engine is developed with the aim of providing the user with tools to facilitate the reassembling of 3D digital fractured fragments. The assisting tools that compose the puzzling engine include 3D manipulation, reference plane alignment, segmentation, and registration. Furthermore, a Virtual Reality (VR) environment is presented as an alternative matching tool. This allows the user to have an intuitive understanding of the fragments in terms of scale, texture, materials, etc., thus facilitating and speeding up the reassembling process. To show the potential of the proposed tool, the engine is tested by archaeologists not only to puzzle classical stone fragments but also to match deteriorated ancient Egyptian rock tomb blocks.

1. INTRODUCTION

3D digital restoration of fractured fragments has been widely studied since it is a fundamental task that plays a crucial role in the restoration of both heritage monuments and archaeological artifacts. Fracture of tangible heritage can be caused by different reasons: deterioration over time, natural catastrophes, looting activities, historical events, etc. As a result, the common denominator found in fractured fragments is the lack of material that prevents restorers from intuitively joining the fragments and retrieve the original shape of the object. Therefore, experts have resorted to traditional techniques to retrieve tangible clues for puzzling. For instance, hand-measurements to match fragments based on scale, meticulous hand-drawings of contours to easily detect fractured areas, and so on. However, this is a time-consuming labor that requires technical expertise. With the maturity of computer vision and pattern recognition techniques, numerous approaches have been proposed to provide experts with automatic clues to match the fragments. Most of these are based on extracting and matching local geometric properties of the points that form the 3D mesh. Although these approaches show promising results, they struggle to retrieve distinctive points for sets with low overlap. This is the main challenge for most of heritage fragments which have undergone severe damage. Hence, for many real-life applications the matching task is still carried out manually as complex features like decoration style or inscriptions meaning are required to solve the puzzle.

In order to assist experts in the matching process, a puzzling engine to join fractured fragments is presented. This digital platform allows the user to interact with the 3D fragments and retrieve geometric features out of the polygonal mesh. As initial setup, the fragments are loaded into a common virtual reference plane, so that registration is reduced to alignment in two dimensions. After startup, the engine provides the expert with tools to facilitate the manual alignment, more specifically, contour surface extraction, principal curvature detection, and pairwise refining registration. On the one hand, the main contour surface offers clear visualization of the shape of fractured areas. On the other hand, the point-cloud of principal curvatures show the relief, decoration, and damage of the fragments. Moreover, the fine alignment allows to successively join fragments while preventing registration errors. The puzzling engine is tested by archaeologists to match different types of fragments, including high/low relief Wall Decorated Fragments (WDF) and stone objects from (Huang et al., 2006). To showcase the potential application of the approach, the former ancient fragments are puzzled in the virtual environment where the objects were found (See Figure 1).

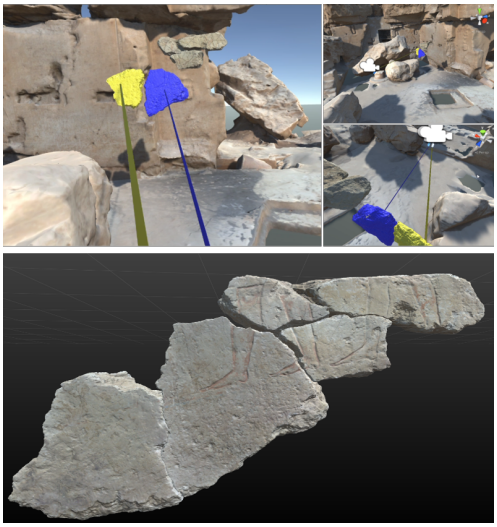


Figure 1: Example of the VR Puzzling Engine, aligning ancient decorated wall fragments.

*Corresponding author

The remainder of the paper is organized as follows. In Section 2, the related work on methods for matching archaeological fragments is presented. In Section 3, the methodology followed to build up the puzzling engine is described. The results and discussion on the performance of the digital platform are presented in Sections 4 and 5. Finally, conclusions and future work are outlined in Section 6..

2. RELATED WORK

In general terms, the task of digitally reassembling fractured fragments boils down to an alignment problem. Typical 3-D registration approaches aim at finding relationships among point-clouds affected by similarity transformations such as translation, rotation and scale. The registration process commonly encompasses three steps: feature extraction, matching, and fine alignment. The first two steps are intended to calculate a rough transformation, while the final one is an iterative process to reduce the distance between point-clouds. The key behind finding a function that relates a point-cloud with its counterpart lies in detecting distinctive characteristics, on the basis of which spatial coherent matches are determined to compute a rigid transformation. Therefore, researchers have focused on developing methods for the extraction of local and global characteristics, based on geometric properties derived from the arrangement of the points in the 3D space. For instance, Huang et al. et al. (Huang et al., 2006) propose an approach to automatically match unassembled objects by locally describing the fragmented region in function of curvatures. Khang Z. et al. (Zhang et al., 2015), tackles the same problem by means of a template-based strategy, in conjunction with normal-based features and curvatures. More recently, by characterizing a fractured surface in terms of curve networks, Mengmeng W. and Jianfeng W. (Wu and Wang, 2018) are able to find trustworthy matches for the reassembling of tiny objects.

The success of automatic reassembling approaches relies on the object completeness. This reduces the complexity of the problem since it is assumed that fractured regions remain intact. However, this assumption does not hold true for heritage fragments that have undergone severe damage and transformations. For instance, fractured surfaces and decoration paint of ancient stone fragments erode away over time, thus preventing existing feature extraction methods from yielding a reliable numerical description of fractured areas. The puzzling of ancient heritage fragments requires, inevitably, the knowledge of experts to identify clues that allow registration techniques to compute an accurate rigid transformation. As a result, computer-assisted technologies have been developed to serve as an empowering tool to facilitate the reassembling process. Kimia B. and Aras H. (Kimia and Aras, 2010) propose an expert-aided interactive platform to assemble vessel fragments, based on the silhouette extracted from images and scans. Benedict B. et al. (Brown et al., 2012) created a tool for visual restoration of fresco fragments, which provides the user with potential alignments merely based on the fragment’s shape and texture.

The presented approach deals with badly preserved ancient decorated wall fragments. Current automatic approaches struggle with this kind of entities due to the lack of matched points. Either local or global features could lead to mismatches since the fractured surface might be incomplete. As for current manual-based strategies, these are not able to cope with erratic shapes and scales, as their matching algorithms assume geometrically-consistent data. The proposed expert-assisted interactive platform takes functionalities of both approaches. On the one hand, various geometric properties are exploited to split the fragment into segments which serve as visual tools to intuitively solve the puzzle. On the other hand, the interface developed enables the user to roughly align the fragments. This task is enhanced by means of an alignment refinement strategy based on ICP. In addition, the platform is instantiated into a VR environment so that user is able to immersively visualize the geometric attributes of the fragments.

3. METHODOLOGY

Figure 2 shows an overview of the steps that encompass the puzzling engine developed on *Unity*. As noted, the workflow is divided in two parts: the starting setup and the puzzling process. The first stage entails computing geometrical features and rendering the respective primitives (plane, contour, principal curvatures). This step is performed only when the fragment is loaded for the very first time, otherwise the vertices of geometric structures are loaded from a serialized file. Once features have been retrieved, the fragment is oriented with respect to a generic plane perpendicular to the z-axis. Since fragments are initially loaded on a 2D imaginary plane, their respective movement is restricted. Translation is carried out only along the x and y axis. Rotation is performed on the basis of Euler angles in degrees along the z-axis. Besides fragment manipulation, the puzzling stage includes pairwise alignment through ICP on the basis of K-d tree nearest neighbor search. After the user feeds the software with an initial rough alignment, the closest neighbor points of the contours are extracted and aligned by using ICP. All fractured fragments are then puzzled by iteratively following the prior process.

3.1 Starting Setup - Extraction of Geometric Primitives

3.1.1 Principal Plane The registration steps discussed in the related work are normally preceded by a segmentation process. This pre-matching step makes total sense as only the fractured area contains vital information for matching. Moreover, the computational burden of the system is notably reduced when splitting the workload into non-dense point-clouds. Segmentation techniques are chosen according to the type of data the problem is dealing with. Since planarity is a prominent characteristic of decorated wall fragments, plane-based segmentation is performed to declutter unmatched areas. This way, the matching task is focused on those points which are most likely to match.

The digital representation of archaeological fractured fragments is by nature noisy and sometimes incomplete. Therefore, segmentation techniques based on local features such as normals (Prakhya et al., 2015) or curvatures (Aldoma et al., 2012), could lead to inconsistent results. However, RANSAC (Schnabel et al., 2007), the well-known method to extract mathematical primitives, has proved to be a reliable approach for filtering data. Its robustness to noise and efficient interpretations make it the ideal candidate to cope with the mentioned constraints.

RANSAC is deployed to extract the main plane of the fragments. Prior to this step, a noise-reduction process is carried out to obtain a trustworthy segmentation. The success of the subsequent steps highly depends on the accuracy of the planar segmentation. On the one hand, the contour extraction is performed on the neighboring points of the plane. On the other hand, the initial transformation is computed by projecting the extracted points on a generic plane. Therefore, two parameters are fundamental to accurately obtain the main plane: *number of iterations* and *distance threshold*. The former parameter depends on the expected outliers ratio.This means that for a noisy dataset a high number of iterations is desirable. The latter parameters are the distance tolerance among the points that form the extracted plane. Higher distance thresholds are expected for bigger models.

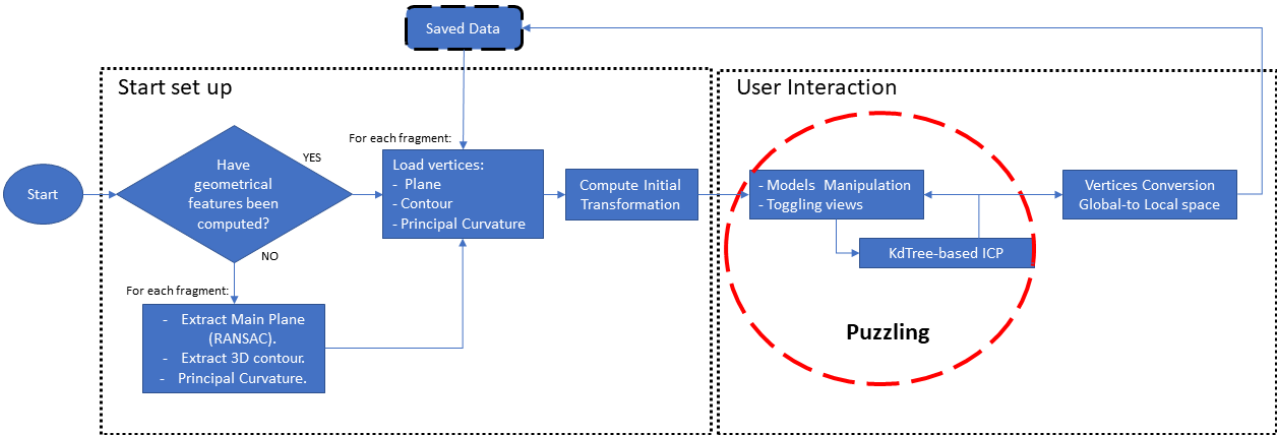


Figure 2: Methodology work-flow for the Puzzling Engine. The starting setup is performed for each fragment of the set. The Puzzling stage refers to pairwise alignment. Both are implemented on the game engine *Unity*.

3.1.2 Contour Surface The Probability of finding matches is higher for the contour points of the plane. By segmenting this area, the user is able to find potential visual clues of possible matches and to perform either automatic or manual alignment. Inspired by the Nearest Neighbor Search (NNS) method (Roussopoulos et al., 1995), our segmentation approach is conducted. NNS aims at finding target points in a given set that are similar to a query point. To this end, different similarity metrics have been proposed, resulting in many NNS variants. For the surface extraction, the Neighbor Distance Ratio (NDR) (Beis and Lowe, 1997) is implemented since we are solely interested in retrieving the closest points with respect to the extracted plane. Considering the plane p and the remaining point-cloud q, the contour surface point-cloud $C(p, q)$ is defined as:

$$C(p,q) = \sum_{0 \le i \le n} \sum_{0 \le j \le m} \text{NDR}(p_i, q_j) \tag{1}$$

where i and j are the indices of the vertices that form the respective point-clouds. The searching ratio r is an important factor of the NDR algorithm. This parameter delineates the region within which the contour points are defined. The contour surface point-cloud becomes denser as the ratio increases. Hence a high r is desirable to obtain a well-defined shape of the contour. To showcase the impact of this value, Figure 3 depicts the contour extracted for different r.

3.1.3 Principal Curvature Decoration of ancient wall fragments is not only based on color but also on sculpture techniques. This holds true for most of the ancient Egyptian fragments, where sunk and raise relief is a common decoration style. In some cases, color fades away and the only remaining distinctive feature is the relief. This kind of decoration is not clearly visible at first glance, the light direction has to be adjusted in order for the relief to stand out. Nevertheless, this characteristic is indispensable for restorers to grasp matching clues. For instance, if the fragment is decorated with lines, by finding the common pattern of linear primitives between two fragments, it is possible to intuitively match them together.

In order to clearly show the decoration of the fragments, we compute the directions and magnitudes of principal surface curvatures (Roussos, 1987) for every point of the fragment . These values contain the necessary information to determine whether a point might belong to a concave or convex surface. The resulting set of points, aids the operator in distinguishing sunk relief without rendering shades. These key-points are computed on the basis of a threshold comparison. Given a point p_i of the extracted plane and its maximum eigenvalue E_{max}, a principal curvature key-point PC_i is defined as follows:

$$PC_i = p_i : E_{max} > Th \tag{2}$$

The principal curvature point-cloud is composed of those points of the extracted plane whose max eigenvalue is greater than a threshold Th. An example of the resulted point-cloud is shown in Figure 4. Note that the principal curvatures key-points show not only the sunk-relief decoration (the feet) but also the fragment damage. This is expected as the shown example contains hammering traces that result in concave surfaces.

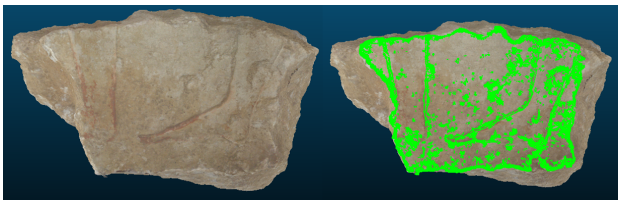


Figure 4: Segmentation of principal curvatures.

3.2 Alignment of Fragments to Generic Plane

In order to facilitate the alignment of planar-based fractured fragments, an initial transformation that relates the decorated surface with a generic plane lying on the world-space is computed. The main idea consists of limiting the fragment's rotational motion to the z axis. To this end, a rigid transformation matrix is calculated as follows. i) A XY plane is defined as: $ax + by + cz + d = 0$ where $a = b = d = 0$ and $z = 1$. ii) Each point of the extracted principal plane is projected onto the XY plane. iii) By using SVD a rigid transformation that relates the extracted plane with the projected plane is computed. This step finalizes the starting setup. From this point onwards the user is able to puzzle by adjusting the position and rotation of the fragments, and the visual assistance of the segmented point-clouds.

3.3 User Interaction - Puzzling

3.3.1 Fragments Manipulation A paramount task when manipulating fragments for puzzling, is the restriction of degrees of freedom (DoF). This simple action tremendously contributes to speed up the alignment process. Since the fragments are initially

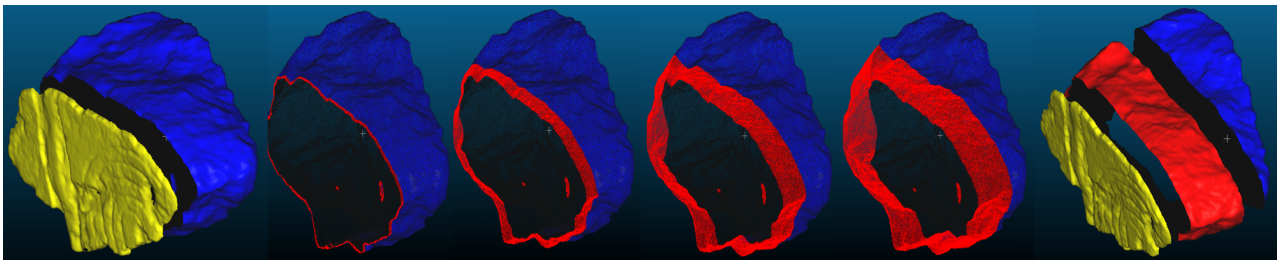


Figure 3: The extreme left models show the extracted plane and its remaining point-cloud marked in yellow and blue respectively. The next five models depict the contour surface point-cloud marked in red, for multiple searching ratios. From left to right, the value of r is, $1.0, 5.0, 10.0$, and 20.0.

instantiated into a generic plane perpendicular to the Z-axis, the fragment's motion is limited to 3 DoF: left/right, up/down and roll. This way the user is able to provide the system with a rough transformation for the pairwise registration step.

A fundamental factor to consider when coping with 3D transformations is the distinction between local and global space. The initial transformation described in the previous section is performed on the local space. This means that each vertex of the point cloud is multiplied by the rigid transformation. The manipulation of fragments, in contrast, is carried out on the global space. Therefore, once a rough alignment is determined, the updated 3-dimensional vectors for rotation and translation are described based on a normalized coordinate system. In order for the fragment's mesh vertices to be independent of the world coordinates, a cube pivot is defined on *Unity* as a partner of the fragment. Principal Component Analysis (Wold et al., 1987) of the respective point-cloud is deployed to define pivot's position and scale. The former corresponds to the centroid and the latter is relative to the principal components.

3.3.2 K-d tree based ICP This step aims at estimating a rigid transformation to improve the rough alignment manually estimated by the user. Since neither matched local points nor common global features are provided, K-NNS (Song and Roussopoulos, 2001) is performed to retrieve the neighbor points between two fragments, on the basis of which ICP is performed to reduce the distance between the clouds. Keeping in mind that the computational complexity of K-NNS relies on the number of points, the contour surface is considered as input rather than the whole point-cloud. K-NNS relies on the k-d tree data structure to efficiently find the nearest point of a set given an input point. Considering a point-cloud Q, a query point p, and the number K of desirable neighbor points. The first step consists of arranging Q into a 3-dimensional binary tree structure. Next, by recursively comparing p and the points in the tree nodes, the algorithm returns K points that exhibit the least square distance with respect to p.

Given two sets P and T roughly aligned, we are interested in retrieving the closest point-clouds P' and T' in order for ICP to estimate a finer alignment. A k-NNS-based strategy is conducted to construct the 3-d arrays that form the sets P' and T'. This process is described step-by-step by Algorithm 1. Basically, for each point p of P, the proposed approach retrieves K neighbors out of T along with their respective square distances. P' is then built by the neighbor points as long as they satisfy two criteria: Firstly, their hamming distance is less than a threshold, thus making sure the found points belong to the nearest vicinity of T. Secondly, they have not been calculated for previous p, preventing P' from redundant points. On the other hand, T' is formed by p_i, if and only if, p has a neighbor point.

Next, a rigid transformation H that finely aligns $P' \rightarrow T'$ is estimated by ICP. Remember, the obtained transformation was determined in function of the global space. Hence, the vertices of P' are transformed to the global space and multiplied by H. The inverse matrix of H is also calculated so that the user is able to undo the automatic alignment.

Algorithm 1 Pseudocode to retrieve the nearest point-clouds between sets P and T

```
procedure CloserPts(P, T)
    Output P', T'
    N ← Size of P
    K ← Number of Neighbors
    Tkd ← KdTree(T)                    ▷ T points into a k-d tree
    th ← Distance Threshold
    for each point p ∈ P do
        flag ← false
        KNNS(p, Tkd, K)                ▷ Computing K-d NN
        NbA ← GetNeighborsArray
        DiA ← GetDistancesArray
        for each element nb, di ∈ NbA, DiA do
            if (di ≤ th) ∧ (nb ∉ T') then
                flag ← true
                WriteIn(P') ← NbArray_id
            end if
        end for
        if flag then
            WriteIn(T') ← p
        end if
    end for
end procedure
```

4. IMPLEMENTATION AND EXPERIMENTS

4.1 User Interface

The User Interface (UI) for the Puzzling engine is developed in Unity, a powerful game engine meant to optimally render high-resolution meshes. This game development software provides a transparent platform to modify mesh attributes: material, vertices, triangles, normals, etc. Our segmentation and registration algorithms are implemented in C++ with PCL Library (Rusu and Cousins, 2011). In order for the algorithmic modules to be compatible with Unity, these were wrapped in a *".dll"* file. As mentioned, the approach is composed of two main modules, starting setup and puzzling. Both are implemented on Unity's environment.

After initialization, the user is able to explore the fragments by means of an orbit camera whose imaginary pivot is located with respect to the cursor's position. Unlike traditional mesh viewers, this digital maneuvering tool enables the user to explore features

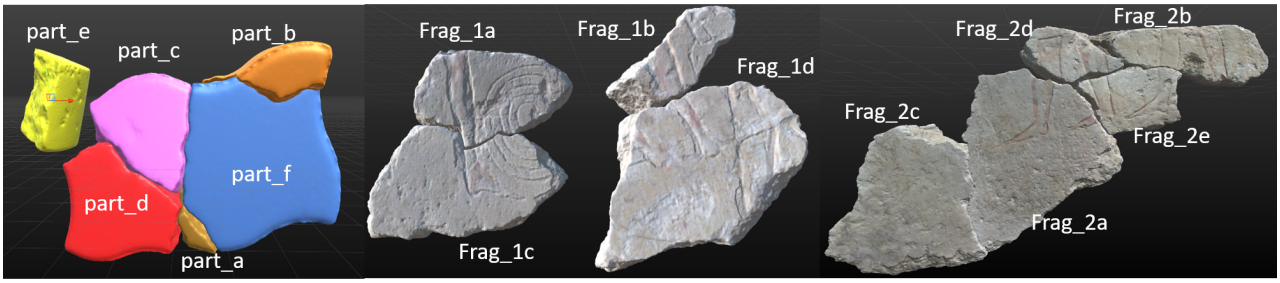


Figure 5: Datasets for experiments after the puzzle is completed.

of multiple fragments at different angles. By means of a 2-D rotation and translation virtual pivot, the operator is able to alter the position of the fragments. Once two fragments are roughly aligned the registration button is activated so that the expert may refine the alignment. When the puzzle is completed, the translation and rotation vector of the *cube collider* as well as vertices of the geometric primitives (local space), are saved and serialized in a binary file. Thus, the next time the software is initialized these values are deserialized, automatically placing the fragments in their correct position.

4.2 VR Implementation

In order to exploit the digital tools that become attainable when designing a VR-based UI, the proposed algorithmic components and the fragments are loaded into a VR environment. The principal approach remains the same, however, the user-machine interaction is enhanced by incorporating virtual components of a real scenario. For example, a scale 3D mesh of the ancient tombs (de Lima and Vergauwen, 2018) where the fragments were found is instantiated as the ambiance for puzzling. Likewise, the decorated wall fragments are scaled-down with respect to the site. This immersive interface not only encourages users to solve the puzzle but also provides them with an extra matching clue: the fragment's scale. This is a tremendously helpful visual component since it allows for size-based categorization of the fragments. For the handling of each fragment, *Oculus touch controllers* are used. A digital infinite ray-line emanated from each hand serves as a virtual indication to select the fragments. Once an entity is selected, rotation and translation movements are defined according to the buttons listed in Table 1. Unlike the standard UI, the VR environment allows the user to maneuver two fragments at the same time. This locomotion flexibility along with the immersive experience drastically reduces the tediousness task of joining fragments.

Button	Action
Button.One(A)or(X)	Rotation left
Button.Two(B)or(Y)	Rotation right
Button.One(A)	Rotation left
Axis2D.Primary/Secondary Thumbstick	Translation (X,Y)
Axis1D.PrimaryIndexTrigger	Toggle model views
Axis1D.SecondaryIndexTrigger	Register ICP
Axis1D.Primary/Secondary HandTrigger	Enable ray-line selector

Table 1: Buttons from fragment's locomotion on the VR environment

4.3 Experiments

The proposed approach was tested on the following device: processor i7-7700HQ at 2.8GHz., 16GB RAM, graphics card GForce GTX 1050. Two datasets of Wall Decorated Fragments(DWF) are considered for the assessment. These belong to excavations of a historical site in Egypt whose development time dates back to the Middle Kingdom period. The fragments are terribly preserved, so it is not advisable to physically puzzle them. The digitization techniques deployed to create the mesh models of both datasets are described by Bassier et al. (Bassier et al., 2018). In addition, the *brick* dataset provided by (Huang et al., 2006) is included in the experiments to show that the method might be practical for other types of stone fragments (See Figure 5).

Since the alignment accuracy highly depends on the operator's expertise, it is not feasible to asses the method based on either registration or matching error. This definitely could lead to an ambiguous discussion and unfair comparison with similar works. The experiments aim to evaluate the interaction user-machine in terms of the software smoothness. This is inferred by calculating the processing times for each algorithmic module. Fast algorithms are desirable to obtain a real-time performance, preventing the puzzling engine from the lagging effect.

As mentioned the performance of algorithmic modules highly depends on their respective parameters. For RANSAC, 1000 is the maximum *number of iterations* selected to extract the main plane of Egyptian fragments. This value is considered due to the inconsistent shape of the fragments, which reduce the probability of extracting a plane. For the brick dataset, on the other side, iterations are limited to 500 as the probability of parameterizing a plane is higher. For both the *distance threshold* is $0.17cm$. For contour surface extraction, the *NNS ratio* is $10-20cm$ depending on the fragment's size. For K-d tree ICP, the *distance threshold* (th) is $30cm$ and $K = 2$. This means that as long as the roughly aligned point-clouds are separated from each other by $30cm.$, the method will be able to refine the alignment. Otherwise, the user will be requested to improve the manual registration.

Processing times of the starting setup stage are listed in Table 2. The segmentation column indicates the time consumed to compute the vertices of the main plane, contour, and principal curvatures. The steer to plane section involves the computational burden for the rigid transformation calculation and rendering of geometric primitives. The starting setup for DWF(2) dataset takes longer because the fragment's resolution is higher. However, once geometric primitives have been computed and saved, loading time is solely dependent on the *Steer to Plane* stage, which is less than $1s$.

For the sake of exemplification, two rough alignments for every dataset are considered to construct the table 3. Figure 6 shows the preliminary joints, which were conducted by the user based on the segmented entities. In the context of the variables previously defined in section 3., P and T correspond to the fragments marked in yellow and red respectively. The number of retrieved closest points that form the sets P' and T' is indicated in the next

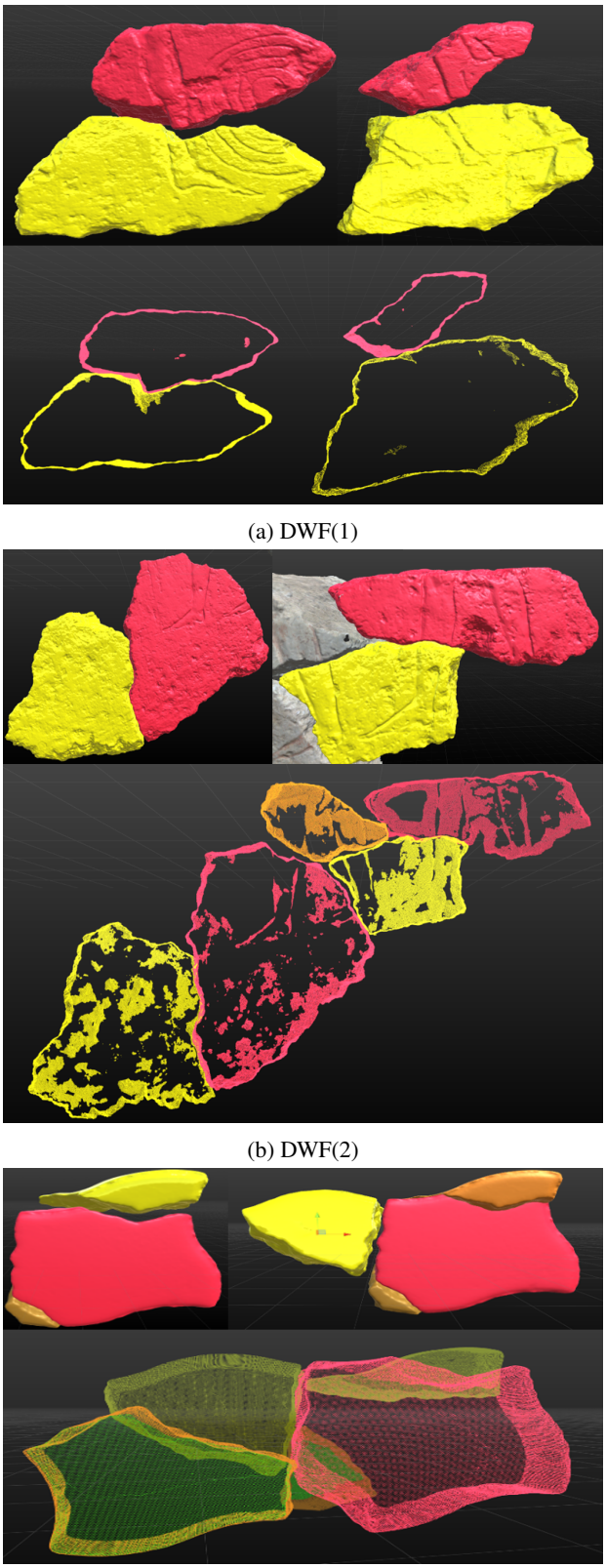
two columns. Unlike prior results, the listed processing times do not follow a linear tendency. This irregular behavior has to do with the fact that the proposed method highly relies on the input alignment. The computational burden for ICP to estimate a rigid transformation is determined by the distance between the fragments. Also, Figure 6 depicts the contour surfaces of the fragments that complete the puzzle. Note that in the DWF(2) dataset, *frag_2d* and *frag_2e*, a fine alignment is not necessary due to the lack of overlap to accurately compute ICP. As for the *brick* dataset the fragment *part_e* is not puzzled since the main plane extracted is not aligned with the other fragments.

Dataset	Fragment	#Points	Segmentation $[ms]$	Steer to Plane$[ms]$
DWF (1)	frag_1a	244446	6549	110
	frag_1b	220158	3906	101
	frag_1c	308832	7541	138
	frag_1d	370217	8121	190
DWF (2)	frag_2a	477503	7063	207
	frag_2b	440351	6698	177
	frag_2c	399448	5335	172
	frag_2d	218446	5178	85
	frag_2e	254529	5953	96
brick	part_a	139420	1901	64
	part_b	169557	2004	69
	part_c	168490	2001	65
	part_d	178721	3165	67
	part_e	189770	3058	72
	part_f	222479	4555	102

Table 2: Processing time for the starting setup.Time to load each dataset: DWF(1)-26.6s,DWF(2)-30.9s, brick-17.1s

Rough Alignment	#Points Query	#Points Target	Alignment $[ms]$
(a) top-left	443	492	1136
(a) top-right	260	270	414
(b) top-left	847	978	1472
(b) top-right	1334	1588	777
(c) top-left	3902	4248	2318
(c) top-right	5170	5664	2372

Table 3: Performance After Post Filtering



(a) DWF(1)

(b) DWF(2)

(c) brick dataset

Figure 6: Rough Alignments for Table 3 (top images). Point-clouds of contour surfaces after fine alignment (button images)

5. DISCUSSION

For Decorated Wall Fragments, the proposed engine served as a practical tool to speed up the manual puzzling. Even though the experimental design is carried out for three datasets, the results give a clear insight into the limitations and potential possibilities of the puzzling engine. On the one hand, for those un-decorated planar fragments with intact fractured surfaces, current automatic or manual approaches are capable of efficiently performing the matching task. An example of this is the *brick* dataset, Huang et al. (Huang et al., 2006) were able to automatically find joints to puzzle the fragments. Whereas by using our engine, the models were partially puzzled. As noted in Figure 5, the fragment *part_e* is not included in the final model since its main plane is not aligned with the front surface of the brick. This result is expected because the dataset fragments are formed by several well-defined planes, so the software aligns the fragment with respect to the plane made of more points. Moreover, since all the fractured surfaces are intact, their silhouette is distinguishable from the fragments alone. Visual primitives, consequently, were seldom used to puzzle this dataset. On the other hand, for those planar fragments that exhibit low overlap and various fractured areas, neither automatic nor semi-automatic approaches are capable of computing trustworthy clues to accurately align the models. The challenge relies on the lack of points to compute distinctive geometric properties. However, the puzzling engine served as a helpful alternative to grasp 3D visual components that effectively aid in the alignment labor of the fragments.

6. CONCLUSION AND FUTURE WORK

In spite of the fact the engine was only tested for a few datasets, the results show that experts might benefit from this technology to puzzle fragments with a high degree of damage. Two aspects were the key to aid in the task of manually finding alignments, the viewer environment flexibility, and the segmented point-clouds. Game engines represent a powerful platform to support the smooth visualization and real-time processing of multiple high resolution 3-D models. Furthermore, they offer the opportunity to extract complex geometric characteristics, empowering experts with visual tools to profoundly study archaeological fragments. The segmented contour and plane resulted in practical ways to find matched areas, while the point-cloud of principal curvatures turned out a functional asset to detect concave surfaces. These might serve as an indicator for professionals to determine properties such as decoration style, damage, hieroglyphic traces, etc. In addition, the VR environment is a complementary tool that enhances the puzzling experience, facilitating the interaction between operator and fragments.

Detection of automatic clues out of the segmented point-clouds is an element considered for future work. Both local and global descriptors could be extracted from the contour surface to automatically provide the user with potential matches. Also, a 2D parametrization of the contour surface will be taken into account to find similarities in terms of concave-convex curvatures. The combination of both approaches could definitely contribute to increase the reliability and automation of the puzzling engine. As for the VR environment, an interaction among the virtual scenario, fragments, and user is also considered for puzzling. Thereby, the user will get automatic clues to digitally place the fragments back to the spot where they originally emanated from.

ACKNOWLEDGEMENTS

The research presented here features within the *Puzzling Tombs* project (nr. 3H170337), funded by the KU Leuven *Bijzonder Onderzoeksfonds*. We acknowledge archaeologists from KU Leuven for actively collaborating in the experiments.

REFERENCES

Aldoma, A., Tombari, F., Rusu, R. B. and Vincze, M., 2012. Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In: *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, Springer, pp. 113–122.

Bassier, M., Vincke, S., de Lima Hernandez, R. and Vergauwen, M., 2018. An overview of innovative heritage deliverables based on remote sensing techniques. *Remote Sensing* 10(10), pp. 1607.

Beis, J. S. and Lowe, D. G., 1997. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: *cvpr*, Vol. 97, Citeseer, p. 1000.

Brown, B., Laken, L., Dutré, P., Van Gool, L., Rusinkiewicz, S. and Weyrich, T., 2012. Tools for virtual reassembly of fresco fragments. *International journal of heritage in the digital era* 1(2), pp. 313–329.

Challis, J. H., 1995. A procedure for determining rigid body transformation parameters. *Journal of biomechanics* 28(6), pp. 733–737.

de Lima, R. and Vergauwen, M., 2018. From tls recoding to vr environment for documentation of the governor's tombs in dayr al-barsha, egypt. In: *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, IEEE, pp. 293–298.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J. and Kwok, N. M., 2016. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision* 116(1), pp. 66–89.

Huang, Q.-X., Flöry, S., Gelfand, N., Hofer, M. and Pottmann, H., 2006. Reassembling fractured objects by geometric matching. *ACM SIGGRAPH 2006 Papers on - SIGGRAPH '06* p. 569.

Kimia, B. and Aras, H. C., 2010. Hindsite: A user-interactive framework for fragment assembly. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, IEEE, pp. 62–69.

Prakhya, S. M., Liu, B. and Lin, W., 2015. B-shot: A binary feature descriptor for fast and efficient keypoint matching on 3d point clouds. In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, IEEE, pp. 1929–1934.

Roussopoulos, N., Kelley, S. and Vincent, F., 1995. Nearest neighbor queries. In: *ACM sigmod record*, Vol. 24number 2, ACM, pp. 71–79.

Roussos, I. M., 1987. Principal-curvature-preserving isometries of surfaces in ordinary space. *Bulletin of the Brazilian Mathematical Society* 18(2), pp. 95–105.

Rusinkiewicz, S. and Levoy, M., 2001. Efficient variants of the icp algorithm. In: *3dim*, IEEE, p. 145.

Rusu, R. B. and Cousins, S., 2011. Point cloud library (pcl). In: *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4.

Schnabel, R., Wahl, R. and Klein, R., 2007. Efficient ransac for point-cloud shape detection. In: *Computer graphics forum*, Vol. 26number 2, Wiley Online Library, pp. 214–226.

Song, Z. and Roussopoulos, N., 2001. K-nearest neighbor search for moving query point. In: *International Symposium on Spatial and Temporal Databases*, Springer, pp. 79–96.

Wold, S., Esbensen, K. and Geladi, P., 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2(1-3), pp. 37–52.

Wu, M. and Wang, J., 2018. Reassembling fractured sand particles using fracture-region matching algorithm. *Powder technology* 338, pp. 55–66.

Zhang, K., Yu, W., Manhein, M., Waggenspack, W. and Li, X., 2015. 3D fragment reassembly using integrated template guidance and fracture-region matching. *Proceedings of the IEEE International Conference on Computer Vision* 2015 Inter, pp. 2138–2146.