Mesh-Based Techniques: A New Approach to Drift-free Natural Navigation in VR

Mark Anson[1] Yangtao Ge[1] Xiaofeng Paul Lin[1] Dao Liu[1] Choi Lam Wong[1] James Zhong[1]

[1]Department of Computer Science, University College London, London, WC1E 6BT UK

***Abstract*—Natural navigation in virtual reality (VR) allows users to travel within the virtual environment (VE) by freely by moving their body and head. However, limited space available to common VR users restricts their ability to explore VEs naturally. Scientists have proposed multiple natural navigation techniques to mitigate the issue, one of the outstanding examples are scale adaptive techniques. However, these techniques would introduce a *Drift* effect. Although correction methods have been proposed in formal studies, none can fully eliminate the drift effect. This paper proposes a new category of techniques called Mesh-Based Techniques (MBTs) and provides four instances of drift-free techniques which achieve natural navigation in relevant areas. The research conducts simulation experiments to justify the viability of MBTs, and our results show that two techniques, Pressure Ring and Lattice Crush, are potentially very effective while the other two perform present issues which would require further improvement.**

I. Introduction

VIRTUAL reality (VR) has grown in popularity as a field of science and has spawned a number of commercial products that are both inexpensive and accessible to non-professional users. Recent advancements in VR hardware technology, such as optical resolution, reductions in movement latency, and screen refresh rate, have catapulted the industry into a new era. Since 2016, most consumer-level VR platforms, such as HTC Vive and Oculus Rift, have been available on the market [1]. The commercial release of VR devices has successfully been embraced in people's everyday lives, with gaming being one of the primary uses [1]. Sutherland proved a definitive display of a room where the user can interact with the object (i.e. a chair or a bullet) in such a display 50 years ago. [2].

Human-Computer Interaction (HCI) scholars have contributed to improving the usability of VR and user experience over the last decade by examining user interfaces that enable users to interact with the scenes, such as manipulations and physical displacement in Virtual Environments (VEs) [3]. In VE exploration, locomotion is recognised as one of the most important facets of interaction to improve the user experience.

We propose a novel navigation approach based on bijective mapping between coordinates in a pair of triangle meshes, which can theoretically eliminate technique-induced Drift. We then devise four distinct techniques to generate meshes: Pressure Ring, Optimization Approach, Lattice Crush and Reverse Lattice Crush. All techniques provide various scaling across the virtual space by moving the vertices of the mesh and adjusting the structural relationship between the triangular meshes.

We design the test environment where 6 flags are evenly placed on the edges of a regular hexagon. The simulation experiment uses the programmable bot to imitate the real VR users. The bot is programmed to move through different flags in order according to the pre-generated test suites.

The paper outlines the potential impacts on user experience adopting the Drift-free navigation techniques and conducts a simulation to evaluate the severity of the negative impacts. The four approaches are compared with the data collected from the simulations. It was found that, Pressure Ring excels at simulating natural navigation in predefined areas, while Lattice Crush provides a better overall performance at the cost of scaling precision in those areas. We discuss the restrictions of the experiment and above-mentioned techniques. Then, we conclude the paper with a discussion on limitations and potential future works.

II. Related Work

This paper will review two related research fields, specifically, Natural VR Navigation techniques, and the effect of Drift and correction techniques.

A. Natural Navigation Techniques

Navigation techniques is a vital research field in VR, which are commonly categorized into wayfinding and travel. Wayfinding is defined as the mental cognitive process for a user to find the direction to the target, whereas travel means the operations conducted by the user to proceed to a target, such as a flag in virtual environment (VE) [4], [5]. These two categories hence are commonly used to evaluate a navigation technique's performance.

With the growing number of commercial VR applications, available space in real has become a major constraint for physical displacement in large VE where the situation requires the reality interaction space to be larger or equal to the virtual space in matching with natural one-to-one mapping navigation [6]. During the decade, several navigation methods have been proposed by VR researchers to mitigate the limitation. According to Nilsson et al. [7], the approaches and metaphors are commonly categorized into three divisions: 1) Re-positioning Systems, the method counteract the forward direction movement so that walking movement can be simulated at the static position; 2) Proxy gestures, the method using user gestures as the a proxy for actual steps and 3) Redirected walking, the method manipulating the actual movement and maps it into virtual movement with slight changes.

Re-positioning methods often require hardware support such as treadmills [8] and human-sized hamster balls [9]. These methods essentially reverse the forward displacement and keep the users static at a certain position which provide a close simulation to natural locomotion. However, these methods can potentially require a large budget to use, and it is complicated to fix the size of interaction space for a home-based user. Also, Bowman et al. demonstrated that re-positioning techniques introduce unnatural movements which are different from walking normally on the ground and hence might cause cybersickness [4].

Walking In Place (WIP) is a commonly referenced proxy gesture technique, which instead of using real physical displacement uses proxy gestures such as swinging arms [10] and stepping feet [11] to represent walking in the virtual environment. Although WIP provides users with a natural and less fatiguing experience when navigating [10], it cannot provide the sense of presence as natural walking, specifically, users cannot perceive where they are when they start or stop moving [12].

Redirected walking (RDW) is a technique to use visual dominance of the user in VE to map physical displacements was first proposed by Razzaque et al. [13]. Generally, RDW manipulates the locomotion path in real to optimize the space by scaling translation, rotation angle, curvature gain and bending gain [7]. However, conventional redirected walking methods only alleviate the space limitation down to 12x12m [14] and 9x9m [15] for square size, which is still restrict the application scenarios, for example, non-professional users and home-based users. To mitigate the issue further, methods called scaled adaptive techniques (SATs) are proposed by adopting the translation gain in RDW and controlling the viewpoint scale and velocity [16] [5]. These methods allow users to move naturally (at 1:1 scaling) at highly relevant areas or "points of interest" and at increased scaling factors in less relevant areas. There are several variants of SATs such as optical flow, head motion and target destination [6].

Based on the previous experiments, SATs usually generate a strong performance in terms of available space, for instance, 3x3m [5] and 5x5 [17]. However, the scale adaptive techniques inherited from dynamic control of viewpoint often produce a side effect called Drift as the paper will discuss in the next subsection.

B. Drift Effect and Correction Techniques

Drift effect can be categorized into two main divisions in terms of its sources, specifically, hardware-produced and techniques introduced [6]. Hardware-based drift is usually related to internal sensors such as gyroscope and motion stereos, the sensor are not accurate enough to capture users' insignificant physical displacement, and accumulation of these implicit drift error contributes to significant deviations for mapping the real interactive environment [18].

The other class of drift effect is often introduced by displacement manipulating navigation techniques such as redirected walking and scaled adaptive techniques, which dynamically modify the user's velocity (i.e. angle and speed) from various scaling factor. The *NaviFields* technique proposed by Montano et al. [5] introduces the drift effect at both experimental level and theoretical level to show the impact and cause of the drift.

A more detailed drift effect analysis is conducted by Montano et al [6]. where it provides a model of uncorrected SATs to demonstrate the mathematics behind the drift effect. Then it proposes a drift correction metaphor based on the the model and two experiments, simulation and user-study respectively, are executed to show the effectiveness of the drift-correction technique. However, it does not fully eliminate the drift effect and it will be exaggerated when adopted to smaller size VEs. Hence, it is worth investigating and proposing a category of drift-free navigation techniques, which is the aim for this paper.

III. Research Hypothesis

A. Navigation Techniques

A potential approach to eliminate drift is to establish a bijective mapping between real and virtual 2D coordinates. With bijective mapping, any real coordinate is mapped to an equivalent virtual coordinate, therefore, the drift defined by Montano et al. [6] is eliminated. Homogeneous scaling is implicitly a bijective map, however it has a constant scaling factor. In order to enable variable scaling factors with bijective mapping, we propose a SAT based on bijective mapping between a pair of triangle meshes using barycentric coordinates. This novel method employs a real mesh covering the available tracking space and a virtual mesh covering the reachable virtual space.

A triangle mesh can be represented by a set of vertices $V \subset \mathbb{R}^2$ and a list of tuples where each tuple is $t = (i, j, k) \in T$ representing a triangle consisting of vertices v_i, v_j and v_k. Given a virtual mesh $M_v = (V_v, T_v)$, the real mesh is $M_r = (V_r, T_r)$ where T_r is identical to T_v. V_r is a result of transforming V_v, $v_i \in V_v$ which corresponds to $v'_i \in V_r$ and $v'_i = Transform(v_i)$ for some transformation function.

To map the real position to virtual coordinates, the barycentric coordinates (α, β, γ) representing the real position with respect to the triangle with vertices v'_i, v'_j and v'_k containing it is first found, then the virtual position is given by $\alpha v_i + \beta v_j + \gamma v_k$. This method offers a way to establish bijective mapping between a triangle mesh pair. As for the transformation, one of the two goals is to enable navigation in a larger virtual space from limited real space, therefore, the transformation is a process of compressing the virtual mesh in order to reduce its size. For the case of homogeneous scaling of scaling factor k, the transformation of each vertex is simply a linear transformation $Transform(v) = \frac{1}{k} I v$, where $I \in \mathbb{R}^{2\times 2}$ is the identity matrix. However, in our case, a more delicate transformation is required so that the triangles within certain areas remain unchanged (in terms of orientation, area and shape) while others are compressed to make space.

There three potential issues of applying the above mapping method on navigation. Firstly, the spatial relationship is not maintained by non-linear transformation which can potentially produce an angle error between the displacements in real space and in virtual space. Secondly, there is a discrete change of scaling and aforementioned angle error when entering another triangle. Lastly, the scaling experienced in different directions could be different given the same position. The first issue could have a direct detrimental effect on user experience because the direction of movement in virtual space is not identical to that in real space. If the angle error is kept below some threshold [19], then the impact on user experience can be minimized. As for the discrete change between two triangles, continuous change can be simulated if enough triangles are used and the triangles are small enough. The impact of the third issue is to be evaluated by experiment.

B. Mesh-Based Techniques

Based on the approach discussed above, this paper proposes a category of methods called Mesh-Based Techniques (MBTs) for VR navigation. MBTs are defined as a process of compressing the size of virtual space to smaller real space, and triangulating both spaces into meshes consist of multiple small triangles, where each triangle follows the barycentric rules. Hence it maintains the one-to-one mapping between virtual and real spaces. In the following subsections, the paper will introduce four instances of MBT, which are Lattice Crush, Pressure Ring, Optimization Approach and Reverse Lattice Crush.

1) Lattice Crush

One such method that utilises this mesh manipulation technique is Lattice Crush. Lattice Crush focuses on heuristically modelling the crushing of a lattice comprised of linear springs of varying spring constants. By continually calculating resultant forces on nodes, and “nudging” them in the direction of the force vector, we can generate meshes that expand around areas of “higher relevance”. This not only avoids the drift problem but can be infinitely customised and fine-tuned depending on the specific environment it is being used for.

Creating the crushed mesh consists of a continuous iterative process, first calculating resultant vector forces of each node in the mesh, then moving the node a small amount in the direction of the resultant vector. This acts to approximate the physics of mesh crushing without creating huge overhead. To implement this technique, the mesh could be represented in many different ways. During research we utilised a Python NetworkX [20] object with each edge containing a ”weight” attribute for the spring constant. Initially, the mesh is shrunk linearly to the desired size, before force calculation and nudging are continually applied.

Force calculations are achieved through a series of iterative processes. Firstly a set of all force vectors from each connected edge at each node is created, calculated from the linear spring equation, $F = kx$. Next the resultant vector from each of those sets of vectors can be calculated using the head-to-tail [21] method, and then the node can be nudged slightly in that direction by finding a new set of coordinates some small distance from the node origin [22].

The quantity by which the node is moved on each iteration is arbitrary, in testing it was set to 0.001 multiplied by the force of the resultant vector for 100 iterations, further fine tuning was then achieved by changing the maximum weight of the edges.

In order to find the weights of each edge, Lattice Crush uses a ”relevance map”, which is also used by Reverse Lattice Crush. For each edge, the coordinates of the midpoint are mapped onto the relevance map to determine the relevancy. During development an image was used to represent this, with opacity as the factor used, although there are many other ways to implement such a solution.

$$weight = \frac{256 - opacity}{255} \times weight_{max} \tag{1}$$

$weight_{max}$ represents the weight applied to an edge in an area of maximum relevance. This value must be fine tuned to produce the desired level of compression.

2) Reverse Lattice Crush

Similar to Lattice Crush, this method nudges vertices around areas of high relevance, but in contrast this method’s virtual mesh starts from a mesh with varying levels of density rather than an uniform lattice. The virtual mesh is then optimized into a uniform mesh and scaled down to the dimensions of the real environment. This technique builds upon readily available mesh generators and optimizers, and avoids lengthy manual mesh manipulation as opposed to Lattice Crush.

Generating a virtual mesh of varying levels of triangle density requires a complimentary relevance map similar to the relevance maps seen in NaviFields [5]. Similarly to Lattice Crush, the relevance map is a grayscale image that represents the virtual environment. White pixels in the relevance map correspond to areas of low interest, darker pixels correspond to areas of increasingly high relevance, and black pixels correspond to areas that should be mapped 1:1 to the physical environment.

The virtual mesh generation step reads in the relevance map and varies the length of edges depending on the gray-scale value at the mapped position of the edge in the relevance map. The virtual environment is first split into a large number of small areas. For each area, their average coordinate is computed and a matching pixel brightness is read from the relevance map file by mapping the virtual environment’s coordinates to the relevance map. From that pixel brightness, an edge length for that area is computed.

Assume an 8-bit grayscale image:

$$L_{edge} = (1 + \frac{p_{brightness}}{255}) \times L_{base} \tag{2}$$

Where L_{edge} is the computed VE edge length, L_{base} a base parameter defining the uniform mesh edge length, and $p_{brightness}$ the brightness of the pixel located at the coordinates mapped from an edge’s coordinates from the VE. The shorter edges occur at the dark areas of the relevance map and result in smaller triangles.

The optimization step then moves vertices around such that all triangles in the mesh reach an uniform size; smaller triangles get stretched while larger triangles shrink. When optimization reaches a certain threshold on triangle area deviation, the whole optimized mesh is uniformly scaled down into the real environment mesh.

3) Pressure Ring

Inspired by NaviFields [5], the Pressure Ring utilizes pairs of regular polygons (rings) to manipulate the mesh. A formal description of a pressure ring is given by a tuple (c, R_{in}, R_{out}) where $c \in \mathbb{R}^2$ is the center, R_{in} is the circumradius of the inner ring and R_{out} is the circumradius of the outer ring. Assuming scaling factor k is desired outside of the natural navigation areas. The transformation function $Transform$ is such that:

$$v' = Transform(v) =$$

$$\begin{cases} \frac{1}{k}Iv, & \text{if } ||v-c|| \geq R_{out} \\ \frac{1}{k}Ic + (v-c), & \text{if } ||v-c|| \leq R_{in} \\ [(||v-c|| - R_{in})\frac{R_{out}/k - R_{in}}{R_{out}-R_{in}} + R_{in}] + \frac{1}{k}Ic & \text{otherwise} \end{cases} \tag{3}$$

Pressure Ring applies linear transformation to vertices outside of the outer ring, however for the area within the inner ring remains unchanged. One beneficial property of Pressure Ring is that, the navigation outside or inside of both rings are theoretically equivalent to homogeneous scaling, the former is of scaling factor k and the latter is of scaling factor 1 (i.e. natural navigation). The issue of angle error does not occur within these two areas.

4) Optimization Approach

The optimization approach of compressing meshes defines a cost function based on the angles and areas of the triangle constituting the mesh and uses simulated annealing to minimize the cost. The cost function consists of two types of costs. The first type, area cost, penalizes transformed triangles having different areas other than the objective areas. The second type, angle cost, penalizes angle differences between interior angles of the original triangle and the transformed one.

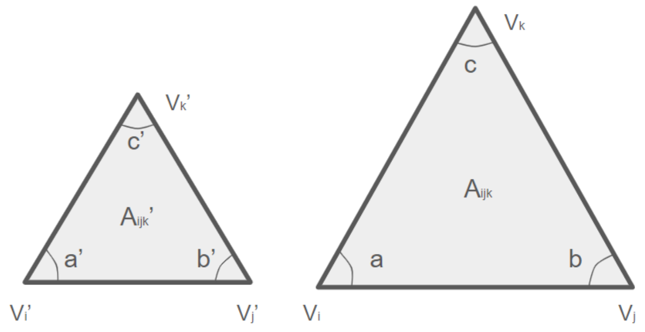


Fig. 1: Illustration for area cost and angle cost

$$AreaCost(t) = AreaWeight_t(\overrightarrow{V_i'V_j'} \times \overrightarrow{V_i'V_k'} - ObjArea_t)^2 \tag{4}$$

$$AngleCost(t) = AngleWeight_t[(a'-a)^2 + (b'-b)^2 + (c'-c)^2] \tag{5}$$

$$Cost(M_v, M_r) = \sum_{t \in T} AreaCost(t) + \sum_{t \in T} AngleCost(t) \tag{6}$$

$$M_r = Transform(M_v) = \text{argmin}_M Cost(M_v, M) \tag{7}$$

$AreaCost$ is utilized with the aim of providing different scaling factor across the virtual space. Assuming the scaling factor k is desired in the region covered by triangle t, $ObjArea_t$ can be set to $k^{-2}A_ijk$ to optimize towards such goal. $AngleCost$ is used to reduce maximum angle error across the entire virtual space by spreading the error. Finally, $AreaWeight$ and $AngleWeight$ specify how much to penalize for angle error or mismatched scaling factor in any given triangle.

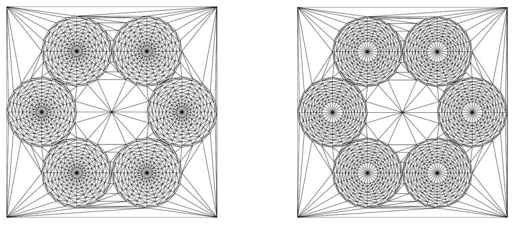
IV. Experiment Design

A. Methodology

1) Test Meshes

Mesh generation with Pressure Ring was based on a specialized mesh where the relevant areas were encircled by 10 concentric "rings" simulated by regular 20-sided polygon, the part outside was triangulation of the vertices of the boundary, the center point and the outermost "ring". The circumradius of the rings in virtual mesh were 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9, 2.0, whereas the vertices of these rings in real rings were transformed by the Eqn.3.

Optimization Approach was based on a triangle mesh grid consisted of triangles (Fig.3a), the height and width of the isosceles triangles were both set to 0.25 and the ones on the boundary were halved. In order to reduce the difficulty of optimization, we expanded the triangles covered by relevant areas and kept the others unchanged, then the entire generated mesh is scaled down by 2 in each dimension. The $ObjArea$ of triangles covered completely by the relevant regions were set to 4 times of its original area, whereas irrelevant triangles were set to its original area. The $AreaWeight$ and $AngleWeight$ were set to 4000 and 1 universally.



(a) Virtual Mesh (b) Real Mesh

Fig. 2: Pressure Ring

Mesh generation for Lattice Crush uses the same triangle mesh grid (Fig.3a) as Optimization Approach, the mesh is also shrunk by a factor of 2 (Fig.3b), to which the Lattice Crush approach detailed in this report is applied.

2) Test Environment

Due to the ongoing pandemic over the course of our research, we were unable to access VR Labs at UCL to gather experimental data from real users. Therefore in our experiment, we sought to simulate a real user's navigation pattern using a Bot script written in C#, attached to a game object within Unity. We decided to build this Bot in a Unity

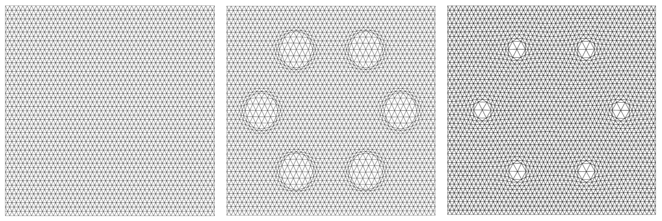

(a) Virtual (b) Real (Lattice Crush) (c) Real (Optimization)

Fig. 3: Lattice Crush and Optimization Meshes

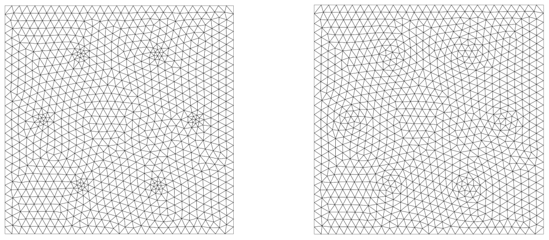

(a) Virtual Mesh (b) Real Mesh

Fig. 4: Reverse Lattice Crush

environment so that if experiments with real users become possible, we can quickly replicate the experiment within the same virtual environment.

The experiment is set up similarly to NaviFields [5], with 6 flags located in the virtual environment (VE), evenly spaced as a regular hexagon and positioned a fixed distance of 4m from the center. We use a scaling factor of 2 between the 6x6m real environment (RE) and 12x12m VE.

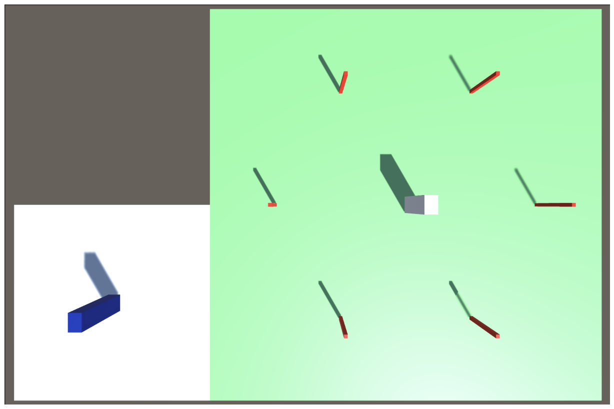


Fig. 5: Real Environment (left) and Virtual Environment (right)

Within this environment setup, we enumerated every path that goes through a 3 and 5 flags sequence, grouped them by test suites of the same optimal path distance, and ran the Bot on each trial. Within each trial, the bot begins from the center of the VE, navigates to 3 or 5 flags in order, and returns to the center of the VE. This ensures that the bot will travel a closed path in each trial, so that we can verify experimentally whether our techniques are Drift-free.

When building the Bot, our aim was to have it behave similarly to real users navigating within the VE. Specifically, the Bot simulates three states of a real user during navigation: accelerating or decelerating when nearby a target, stationary rotation when turning towards the next target, and constant speed walking in between.

The Bot game object is created as a 0.35x0.5m rectangular prism with a height of 1.8m, to approximate the proportions of an adult male. The constant walking speed for the Bot is set to 1.4m/s, corresponding to the walking speed of an average adult [6]. The Bot maintains its position in the RE and VE, which we'll refer to as its real position (RP) and virtual position (VP).

At the start of a trial, RP and VP are both at the center of the RE and VE. Before moving towards it's first target, the Bot begins by orienting itself to the target. It calculates the difference between the target position (TP) and VP, then uses this to compute the target angle [0, 360] in degrees. It computes the difference between the target angle, and the rotation of the Bot (identical in virtual and real), and rotates clockwise or anticlockwise by 5 degrees every 0.02 seconds (or 250 degrees per second), depending on which rotation requires less than 180 degrees to reach the target.

Once the Bot is aligned to the direction of the target in VE, it begins accelerating, with the walking speed of 1.4m/s scaled by a speed scaling factor s capped at 1, computed as $s = 0.1 + d$, where d is the distance from VP to the nearest TP. For example, when the Bot is starting a trial, d = 0, s = 0.1, so the starting speed is $0.1 \times 1.4 = 0.14$m/s. As the trial continues, s will increase up to 1, corresponding to a max speed of 1.4m/s when the Bot's VP is ≥ 0.9m or further from the nearest TP.

Every 0.02s, the RP is translated forwards using the unit forward vector associated with its game object by $v \times s \times t$, where v is the fixed speed of the Bot 1.4m/s, s is the speed scaling factor, and t is the elapsed time between function calls at 0.02s. The VP is updated by translating the new RP to it's coordinates in VE by finding its equivalent encompassing vectors and using barycentric mapping to transform its coordinates from real to virtual.

When the Bot's VP is within 1.1m of the target flag, it begins to decelerate, with s decreasing from 1 to 0.1, corresponding to the distance of 1.1 to 0.2m from the flag. The Bot updates its target to the next flag as soon as it's 0.2m from the flag, and begins the process of orienting itself, accelerating, traveling at fixed speed (in RE), and decelerating again.

B. Data Collection

Meshes for the real world and its corresponding virtual world were generated using each proposed navigation technique, represented in the form of its vertices and indices. These were then serialised into mesh files and then subsequently deserialised in Unity 3D into mesh objects for testing.

Certain variables were measured on the real and virtual bot during the simulation experiments: Completion time (CT), real distance travelled (RD), trajectory deviation (TD) and angle deviation (AD). CT is the time taken for the virtual bot to leave and return back to the centre. TD is the ratio between the total displacement distance of the virtual bot travelled and its optimum path (travelling in linear paths between the each of the bot's stationary points in the relevant areas) and lastly, AD is the absolute angle difference between the direction of the path taken by the real bot and the virtual bot in each frame.

Prior to each simulation, the scale and positioning of the testing environment in Unity were adjusted based on the meshes given and the conditions outlined in each test suite. In each simulation run, the metrics and positional data are recorded for both the real and virtual movements during each run were measured from the bot at a fixed rate of 50 frames per second and written to files for analysis.

C. Metrics

Although the Unity simulator have captured several metrics as above mentioned, the property of the aforementioned simulation determines that CT and RD will be quite similar across different techniques, since the bot is programmed to move along side the certain path in real space and always keep the same speed at each corresponding frame. However, these metrics would become effective in the study involving the actual VR users. According to Nilsson et al. [23], in most of redirected walking user case studies, people would feel cybersickness when experiencing large discrepancies of curvature gains and translation gain.

To evaluate the performance of this specific simulation experiment, this paper primarily focuses on two metrics, specifically, angle differences between real and virtual displacement and scaling factor changing during movements. However, the metrics from Unity alone cannot reflect the user experience directly, hence, two analytic methods are used for processing the draft metrics, which are changing gradient and accumulation effects.

The integration of scaling factor changing over virtual space (ISFCR) representing the accumulation of changing scaling factor and it can be expressed as following: where ΔV and ΔR stands for the displacement in virtual and real space respectively:

$$\sum_{\Delta V} abs(\frac{||\Delta V||}{||\Delta R||} - 1) * ||\Delta V|| \tag{8}$$

It is worth noting that this integration is only calculated within the relevant area, since all scaling navigation techniques will have the same accumulation of scaling factor changing after travelling a certain distance, while the distribution of the scaling factor changing varies. Also, from the purpose of the relevant area, the scaling factor of new methods are expected to be as closed to 1.0 as possible within the relevant area. Correspondingly, the smaller ISFCR represents a better performance of this technique so that the users could experience more natural navigation within the relevant area.

For the other perspective of scaling factor, the gradient of scaling factor against virtual distance travelled (GSF) demonstrates how fast the scaling factor is changing and the equation can be written as follows, where dS means the changing of scaling factor, and $d||V||$ means the virtual displacement:

$$\max(\frac{dS}{d||V||}) \tag{9}$$

The maximum value shows the worst case of scaling factor changes, hence, the higher value illustrates a poor performance of the technique. This value reflects the how severe the user could experience the variation of scaling.

Similarly, the analysis adopts the accumulation method for the angle differences (IADA & IADR), where it uses Θ to represent the angle difference between virtual and real displacement in degrees as the equation below shows:

$$\sum_{\Delta V} \Theta * ||\Delta V|| \tag{10}$$

This quantifies the angle error the user could experience when travelling across the entire virtual space. Unlike scaling factor, some of the approaches do not have angle difference such as *NaviField* [5], so that it is worth to evaluate the integration across the entire virtual space. In the benchmark, we evaluate the angle differences both within the relevant area and across the entire virtual distance travelled.

The gradient of angle differences (GAD) versus virtual distance travelled is also a vital metric, which represents how fast the angle would change within a certain virtual distance. The equation is written as:

$$\max(\frac{d\Theta}{d||V||}) \tag{11}$$

Similar to GSF, the paper takes the maximum to represent the worst case, consequently, the higher value suggests a mediocre result and this value illustrates the severity in the variation of angle differences.

D. Benchmarks

To assess and evaluate the performance and effectiveness across above-proposed 4 techniques, this article would first analyse NaviFields which is the baseline for the evaluation to illustrate the advantage of the MBTs in this paper. The motivation of the paper indicates the research aims to propose a metaphor where it is Drift-free and maintains 1-to-1 mapping within the relevant area.

Previous study on NaviFields and Drift Correction method show that a noticeable effect of 0.6 metre and 0.2 metre for with and without drift-corrected methods respectively, hence the benchmark after would demonstrate the drift gain for each method. Also, by definition of NaviFields the relevant area is defined as 1-to-1 mapping, i.e. natural navigation, so that the ISFCR can be regarded as 0 within random error. Whereas techniques proposed in this paper require further analysis on the benchmark.

Figure 6-10 provides an overview of the results obtained from this research, showing the final average and standard deviation of the five metrics mentioned above i.e. ISFCR, GSF, IADR, IADA and GAD for each test suite. These test suites are ordered by the distance travelled in virtual space and the number of flags the bot visited.

In the research hypothesis, the paper has demonstrated that all MBTs aim to eliminate drift effects when the user travels across the virtual environment. This paper compares the coordinates between where the bot begins and where the bot ends and calculates the distance between the two points,

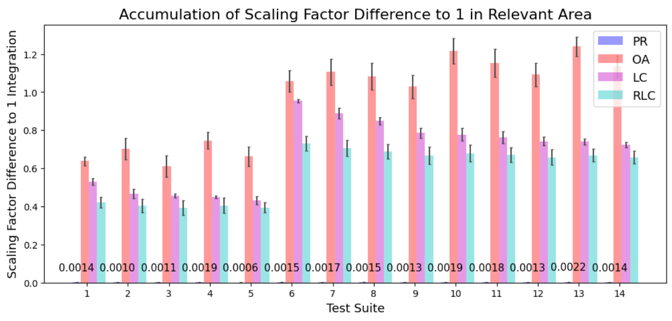


Fig. 6: The bar chart demonstrating the accumulation effect of scaling factor differences to scaling factor of 1 within the relevant area (ISFCR).

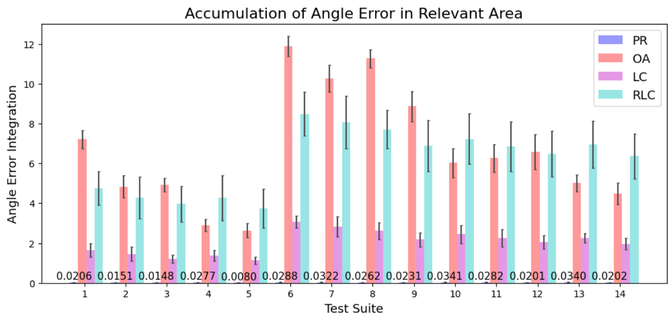


Fig. 7: The bar chart demonstrating the accumulation effect of angle error to within the relevant area (IADR).

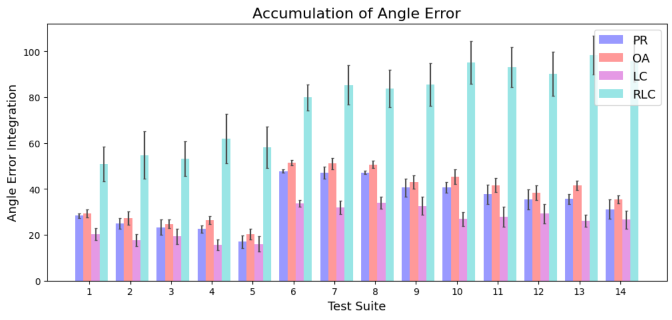


Fig. 8: The bar chart demonstrating the accumulation effect of angle error across the whole path the bot travelled (IADA).

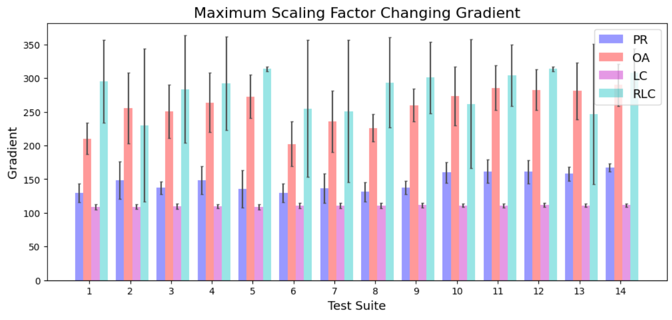


Fig. 9: The bar chart demonstrating the maximum gradient of scaling factor changing across the whole path the bot travelled (GSF).

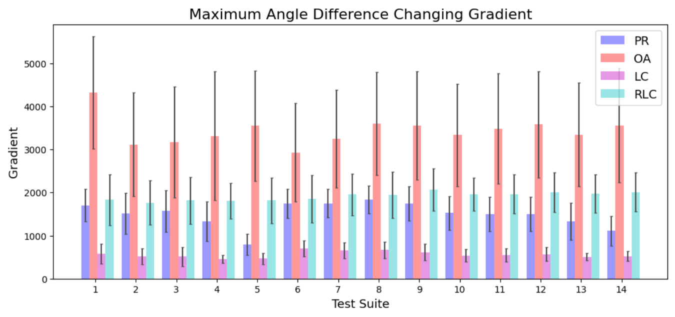


Fig. 10: The bar chart demonstrating the maximum gradient of angle difference changing across the whole path the bot travelled (GAD).

the drift gain [6] is at the magnitude of 10^{-3} metre for every technique proposed in the paper. Hence, our experiment shows that for each test trial, the bot could return back to the origin where it starts from within the random error.

Besides the drift effect, this paper evaluates the natural navigation within the relevant area by examining the accumulation changing error within the relevant area as figure 6 and 7 demonstrate. This metric is used for demonstrating the naturalness of travelling in the relevant area, where the low value suggests that it is close to natural navigation.

Additionally, it analyses the angle error across the entire virtual distance travelled, where the accumulation effects are plotted in figure 8 and along with two more metrics, maximum gradient of angle difference and scaling factor difference to 1 respectively, are adopted to quantify the overall experience of the bot. These three metrics indicate how VR users might perceive their navigation through the environment during the whole trial.

V. Analysis of Results

In this section, we present and analyse the result of the bot simulation on meshes generated by different techniques (Pressure Ring (PR), Optimization Approach (OA), Lattice Crush (LC), Reverse Lattice Crush (RLC)). The means and standard deviations of the aforementioned metrics of 16 different test suites are plotted. The test suites are numbered such that the bot is given 3 targets in test suites No.1-5 and 5 targets in test suites No.6-14. In addition, within the two groups of test suites, they are ordered in an increasing order of length of optimal path. The primary concern here is to compare different mesh manipulation techniques with respect to their performance on distortion and scaling.

In terms of the performance of navigating in the relevant area (Fig.6,7), Pressure Ring is almost identical to natural navigation and has a significantly better result than all the others. A clear pattern of accumulated scaling difference is shown (Fig.6). Lattice Crush came second, Reverse Lattice Crush was third, finally Optimisation approach was the worst. Intriguingly, Lattice Crush's performance seemingly improves with an increased length of the optimal path. As for accumulated angle error (Fig.7), the performance of Optimization Approach and Reverse Lattice Crush are worse than that of Lattice Crush with substantially more angle error.

Angle error outside of the relevant area is critical as well. Accumulation of angle error over the entire lifetime of the bot (Fig.8) indicates that the bot experienced remarkably more distortion than the others. In contrast, Lattice Crush has the least distortion; the distortion of Pressure Ring and Optimization Approach are similar and are both slightly greater than Lattice Crush; Reverse Lattice Crush is by far the worst performer.

The maximum scaling gradient (Fig.9) and the maximum angle difference gradient (Fig.10) account for the changes of scaling and angle difference. Lattice Crush performs the best in both measures, followed by Pressure Ring; Optimization Approach and finally Reverse Lattice Crush.

In conclusion, if the performance in relevant area is the priority, then Pressure Ring could be the suitable technique to apply. However, while sacrificing performance in the relevant areas, Lattice Crush provides the best overall performance across the entire virtual space. Reverse Lattice Crush produces the most angle error and Optimization Approach does not perform well both inside and outside of the relevant area, so, unless they are improved in the future, these two mesh manipulation techniques are not suitable for mesh-based navigation.

VI. Discussion and Limitations

Our analysis produced a few key results. Both Lattice Crush and Pressure Ring come out as having good potential for further review. Pressure Ring is very good at achieving 1:1 scaling with reasonable changes in scaling to the less relevant areas, whereas Lattice Crush minimises angle error. We believe that with further research, either of these techniques could present a viable new navigation technique. This does not discount either the Optimisation Approach or Reverse Lattice Crush, as it is possible that with further tweaks or with different experimental design these techniques may reveal advantages not exposed in this report.

The effectiveness in both perceptibly and effective scaling of all discussed Mesh-Based Techniques will inevitably depend heavily on the exact layout of the virtual environment used. Similarly to NaviFields, the techniques will be most effective when environments have areas of varying degrees of relevancy. Without this variance these techniques will perform very similarly to homogeneous scaling. However Lattice Crush in particular will benefit from a careful application of scaling due to issues regarding borders between areas of high and low relevance, discussed later in this section.

In the NaviFields paper, it is suggested that the navigation field used in the technique could be initially homogeneous and then adapted through clustering techniques as the user moves through the environment. Such an implementation could potentially be possible with Lattice Crush, as the weights of edges could be increased or decreased depending on how much time the user spends there. This may also be possible with Pressure Ring by adding rings to areas the user spends the most time in.

Some of the most important limitations with our experiment relate to the Bot. Although the Bot provides a coarse simulation of a real user's navigation, user experience is very important when evaluating the performance of a navigation technique. Subjective metrics such as Ease, Comfort [5], Control, Natural [6] cannot be measured by a Bot, but is an important part of evaluating any navigation technique. Although some predictions can be made based on a visual assessment during the trials or analysis of the data, we do not know for certain how users will perceive these techniques.

Additionally, the real-world Bot moved under the assumption that it could instantly process the direction to take at all times, when in reality the distortions in the mapping can affect the spatial perception of the user. Not having access to a headset also prevented us from investigating the effect of our meshes on manoeuvring. Perhaps one upside of using the bot versus a headset is that the measured TD was unaffected by the lateral movements produced if we were to use a VR headset, an issue mentioned by Montano et al. [5]; our calculated value was only concerned with the movement of the torso.

Besides the Bot, there are further limitations with our Experiment Design. Firstly, although we've set up the test environment in a similar manner to NaviFields [5], we cannot do a direct comparison since we selected a different sized Real Environment (RE). In the NaviFields paper, a 3x3m RE is used, but we're simulating our Bot within a 6x6m RE, due to constraints imposed by Pressure Ring.

In addition, we've only conducted tests for a scaling factor of 2 between VE and RE. Therefore, it remains unclear how performance would be impacted for other mesh generation techniques if we raise this scaling factor to 4 or 8.

This experiment used NaviFields as the baseline, however it was not included in our simulations. Hence, we do not have an equivalent data-set for NaviFields from our experiments. Although we can see the relative performance of using different mesh generation techniques, it's less clear how each technique performs against other Scale Adaptive navigation techniques previously mentioned.

The key limitation of Lattice Crush is the inability to achieve the large changes in scaling across a mesh that Pressure Ring is able to do. During earlier testing we found that as we produced scaling closer to 1:1 in the relevant areas, the boundary between the relevant and less relevant areas became much more pronounced, to the point where we saw the virtual bot teleport across the boundary. This means that for our particular test setup we are unable to achieve natural walking (1:1 scaling) within the highly relevant areas as was demonstrated in the experiment.

An issue with Reverse Lattice Crush is that the optimization step is a global operation. The entire mesh is moved around when stretching out the denser parts of the virtual mesh. While this is unsurprising for the points of interest, it also creates both angular and scaling distortions in parts of the mesh that should have been homogeneously scaled. The distortion produced is reflected by having the largest accumulation of angle error among all techniques (Fig.8). Furthermore, because the triangle density of the VE varies depending on the relevance map, the factor to downscale the optimized mesh by to reach 1:1 scaling factor in areas of high relevance is not constant. This makes reaching 1:1 scaling factor difficult and often inexact as a result. Regardless, Reverse Lattice Crush still has a decent performance in terms of scaling inside the relevant area (Fig.6).

Despite being able to provide interaction much similar to natural navigation in the relevant area, Pressure Ring imposes extra constraints on the design of the virtual space. For instance, assuming the scaling factor of k is to be adopted, the constraint is that $kR_{in} < R_{out}$. The intersection of different pairs of rings is also undefined, consequently, the regions of natural navigation must be separated by certain distances. Also, the total area of these regions inside a given virtual space is limited, and the shape of these regions are limited by a regular polygon. Angle errors exist at coordinates within the set differences of two rings. Pressure Ring also compresses the space in those regions and the average scaling factor is increased, as a result, users traveling towards the center of relevant area will first experience an increase in the scaling followed by a sudden discrete drop to 1, where gradual continuous change is more favourable. The above two drawbacks can be mitigated by increasing R_{out}. The average angle error can be decreased and the difference in scaling factor when entering the rings can also be minimised, but the sudden drop to scaling factor 1 is inevitable with Pressure Ring. However, increasing R_{out} strengthens the design constraints mentioned above and fewer natural navigation areas are allowed. So, there is a trade-off between user experience and design constraints.

Compared to Pressure Ring, the Optimization Approach does not have the aforementioned explicit design constraints, allowing natural navigation regions of more complicated shapes to be implemented and placed close to each other. However, implicit constraints still exist, although the cost function penalizing transformations producing a mesh that has overlapped triangles, optimization can still produce meshes that are undefined for our bijective mapping if the natural navigation regions are set to be too large.

VII. Conclusion and Future Work

In this paper, we explored Mesh-Based Techniques (MBTs), a Drift-free category of natural navigation techniques which uses barycentric coordinates and a pair of real and virtual triangular meshes to establish a bijective mapping between real and virtual spaces in VR. These techniques extend the navigable space in VE as with SATs but avoids technique-induced drift.

We provide the mathematical characterisation and implementation of four different MBTs: Lattice Crush, Reverse Lattice Crush, Pressure Ring, and Optimization Approach. We implemented a test environment and created a Bot to provide a coarse simulation of real user navigation patterns, in order to evaluate performance for each technique using derived metrics when users perform a comprehensive set of 3 and 5 flag travelling tasks.

Analysis of the results show that Pressure Ring is ideal for applications requiring 1:1 scaling within the relevant areas, whereas Lattice Crush performs best in minimising accumulation of angle error, and has a smoother change in scaling factor and angle difference in travelling tasks, providing a better overall performance. We believe with further development and real user testing, MBTs could serve as useful alternatives to existing SATs for extended navigation within small real spaces.

In future work, we plan to conduct the experiment with real users, using a similar design to the one in the Drift Correction Techniques paper [6]. We would collect objective metrics like the ones in our current experiment, and collect a range of subjective impressions such as Comfort, Ease, and Control.

It would also be worth exploring a range of scaling factors to determine how effective these Mesh-Based Techniques are when subject to higher scaling. Under a real user study, a clear limit in the scaling can be found when movement becomes too distorted and therefore not viable to use.

We would also like to conduct a user experiment for Maneuvering Tasks, similar to that in NaviFields [5], which involves tasks that require precise movements. We would be interested to find out how our technique compares with NaviFields when using real users.

In addition, it would be useful to establish a more precise relationship between the subjective user experience and observable objective measures. For example, what magnitude of angle deviation is imperceptible to a user, and what is the threshold where the angle deviation becomes intolerable? Establishing these relationships would enable researchers to build better automated testing and evaluation tools for navigation techniques proposed in the future.

References

[1] M. Al Zayer, P. MacNeilage, and E. Folmer, "Virtual locomotion: a survey," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 6, pp. 2315–2334, 2018.

[2] I. E. Sutherland, "The ultimate display," pp. 506–508, 1965.

[3] J. J. LaViola Jr, E. Kruijff, R. P. McMahan, D. Bowman, and I. P. Poupyrev, *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.

[4] D. A. Bowman, E. Kruijff, J. J. LaViola Jr, and I. Poupyrev, "An introduction to 3-d user interface design," *Presence: Teleoperators & Virtual Environments*, vol. 10, no. 1, pp. 96–108, 2001.

[5] R. A. Montano Murillo, E. Gatti, M. Oliver Segovia, M. Obrist, J. P. Molina Masso, and D. Martinez Plasencia, "Navifields: Relevance fields for adaptive vr navigation," in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, 2017, pp. 747–758.

[6] R. A. Montano-Murillo, P. I. Cornelio-Martinez, S. Subramanian, and D. Martinez-Plasencia, "Drift-correction techniques for scale-adaptive vr navigation," in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, 2019, pp. 1123–1135.

[7] N. C. Nilsson, S. Serafin, F. Steinicke, and R. Nordahl, "Natural walking in virtual reality: A review," *Computers in Entertainment (CIE)*, vol. 16, no. 2, pp. 1–22, 2018.

[8] R. P. Darken, W. R. Cockayne, and D. Carmein, "The omni-directional treadmill: a locomotion device for virtual worlds," in *Proceedings of the 10th annual ACM symposium on User interface software and technology*, 1997, pp. 213–221.

[9] E. Medina, R. Fruland, and S. Weghorst, "Virtusphere: Walking in a human size vr "hamster ball"," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 52, no. 27. SAGE Publications Sage CA: Los Angeles, CA, 2008, pp. 2102–2106.

[10] N. C. Nilsson, S. Serafin, and R. Nordahl, "The perceived naturalness of virtual locomotion methods devoid of explicit leg movements," in *Proceedings of Motion on Games*, 2013, pp. 155–164.

[11] J. Feasel, M. C. Whitton, and J. D. Wendt, "Llcm-wip: Low-latency, continuous-motion walking-in-place," in *2008 IEEE symposium on 3D user interfaces*. IEEE, 2008, pp. 97–104.

[12] J. N. Templeman, P. S. Denbrook, and L. E. Sibert, "Virtual locomotion: Walking in place through virtual environments," *Presence*, vol. 8, no. 6, pp. 598–617, 1999.

[13] S. Razzaque, D. Swapp, M. Slater, M. C. Whitton, and A. Steed, "Redirected walking in place," in *EGVE*, vol. 2, 2002, pp. 123–130.

[14] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas, “Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 555–564, 2012.

[15] T. Grechkin, J. Thomas, M. Azmandian, M. Bolas, and E. Suma, “Revisiting detection thresholds for redirected walking: Combining translation and curvature gains,” in *Proceedings of the ACM Symposium on Applied Perception*, 2016, pp. 113–120.

[16] J. D. Mackinlay, S. K. Card, and G. G. Robertson, “Rapid controlled movement through a virtual 3d workspace,” in *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, 1990, pp. 171–176.

[17] X. Xie, Q. Lin, H. Wu, G. Narasimham, T. P. McNamara, J. Rieser, and B. Bodenheimer, “A system for exploring large virtual environments that combines scaled translational gain and interventions,” in *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, 2010, pp. 65–72.

[18] T. Oskiper, H.-P. Chiu, Z. Zhu, S. Samaresekera, and R. Kumar, “Stable vision-aided navigation for large-area augmented reality,” in *2011 IEEE Virtual Reality Conference*. IEEE, 2011, pp. 63–70.

[19] V. Interrante, B. Ries, and L. Anderson, “Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments,” in *2007 IEEE Symposium on 3D User interfaces*. IEEE, 2007.

[20] “networkx.org,” https://networkx.org/.

[21] “Physics classroom - addition of forces,” https://www.physicsclassroom.com/class/vectors/Lesson-3/Addition-of-Forces.

[22] “Mathematics stack exchange - find a point on a line segment, located at distance https://www.overleaf.com/project/6029865a48e426a18756648cd from one endpoint,” https://math.stackexchange.com/questions/134112/find-a-point-on-a-line-segment-located-at-a-distance-d-from-one-endpoint.

[23] N. C. Nilsson, T. Peck, G. Bruder, E. Hodgson, S. Serafin, M. Whitton, F. Steinicke, and E. S. Rosenberg, “15 years of research on redirected walking in immersive virtual environments,” *IEEE computer graphics and applications*, vol. 38, no. 2, pp. 44–56, 2018.