Dynamic Occlusion Handling for Real-Time AR Applications

Joaquim Jorge
INESC-ID / Técnico / U. Lisboa
Lisboa, Portugal
jorgej@acm.org

Rafael Kuffner dos Anjos
INESC-ID / Técnico / U. Lisboa
Lisboa, Portugal
ranjos@acm.org

Ricardo Silva
Instituto Superior Técnico / U. Lisboa
Lisboa, Portugal
ricardo10silva@gmail.com

ABSTRACT

Augmented reality (AR) allows computer generated graphics to be overlaid in images or video captured by a camera in real time. This technology is often used to enhance perception by providing extra information or simply by enriching the experience of the user. AR offers a significant potential in many applications such as industrial, medical, education and entertainment. However, for AR to achieve the maximum potential and become fully accepted, the real and virtual objects within the user's environment must become seamlessly integrated. Three main types of problems arise when we try to achieve this effect: illumination issues, tracking difficulties and occlusion troubles. In this work we present an algorithm to handle AR occlusions in real time. Our approach uses raw depth information of the scene to realize a rough foreground / background segmentation. We use this information, as well as details from color data to estimate a blending coefficient and combine the virtual objects with the real objects into a single image. After experimenting with different scenes we show that our approach is able to produce consistent and aesthetically pleasing occlusions between virtual and real objects, with a low computational cost. Furthermore, we explore different alternatives to improving the quality of the final results while overcoming limitations of previous methods.

CCS CONCEPTS

• **Computing methodologies → Visibility**; **Mixed / augmented reality**; *Rendering*.

KEYWORDS

Augmented Reality, Real-time Realistic Occlusion, Dynamic Occlusion Handling, Alpha Matting

ACM Reference Format:
Joaquim Jorge, Rafael Kuffner dos Anjos, and Ricardo Silva. 2019. Dynamic Occlusion Handling for Real-Time AR Applications. In *The 17th International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI '19), November 14–16, 2019, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3359997.3365700

1 INTRODUCTION

Augmented reality (AR) is an upcoming technology that allows computer generated graphics to be overlaid in images or video captured by a camera in real time. This technology is often used to enhance perception by providing extra information or simply by enriching the experience of the user. Recent results show that AR offers a significant potential in many applications such as industrial [Schmirler et al. 2018], medical [Lopes and Jorge 2019; Zorzal et al. 2019], education [Herbert et al. 2018; Preim and Saalfeld 2018] and entertainment [Schmidt et al. 2019; Shah et al. 2012].

According to Breen *et al.* [Breen et al. 1995] for AR to achieve the maximum potential and become fully accepted, the real and virtual objects within the user's environment must become seamlessly integrated. Three main types of problems arise when we try to achieve this effect: illumination problems, tracking problems and occlusion problems.

Occlusion occurs when an object closer to the viewer obscures the view of objects further away [Breen et al. 1995]. In most AR applications the virtual objects occlude the real objects but sometimes the opposite can happen. For example, when the virtual object moves behind a real object, which can cause an incorrect occlusion [Zhu et al. 2010]. Research on this problem developed three different approaches: Model-based [Breen et al. 1995], [Fischer et al. 2003], [A. Fortin and Hebert 2006], Object-based [Lepetit and Marie-Odile 2000], [Yuan et al. 2010] and Depth-based [Breen et al. 1995], [M. Wloka and G. Anderson 1995], [Hayashi et al. 2005].

In this work we focus on solving the occlusion problem in a realistic manner. As depth sensors evolve algorithms we can now use depth information to handle occlusions. However, the information provided by the sensors still has some problems: different types of noise, shadow effects and inaccurate mapping between depth and color data.

To improve the depth information researchers have come with multiple ways to improve the depth maps provided by the sensors [Du et al. 2016; Leal et al. 2013; Schmeing and Jiang 2014]. Du *et al.* [Du et al. 2016] proposed a technique where the depth map edges can be aligned with the color map edges. This method only works well if the background and foreground are clearly distinguishable so it still does not provide a robust answer to the problem. On top of that, sometimes fuzzy objects can cause regions or pixels to not explicitly be on the foreground or on the background. In those cases we can get a more realistic result if we determine the relative transparency of the objects instead of just trying to figure out if they should be in front or in the back of each other. So, we can conclude that instead of just trying to find if we should display the virtual objects or the real objects in a per pixel basis a better approach would be to determine a blending coefficient for each pixel, also called alpha matte.

In this work we tackle the occlusion problem as an alpha matting problem and propose a method that is able to produce realistic occlusions in static and dynamic scenes. As a starting point we took inspiration from [Hebborn et al. 2017] and we propose some changes to that algorithm to improve the quality of the results.

This document is organized as follows: Section 1 introduces the problem and describes the objectives of the work. Section 2 describes occlusion as an alpha matting problem. In Section 3 we discuss the related work about this subject. In Section 4 we describe the overview of the proposed approach. Section 5 to Section 10 describe the method in detail. Section 11 describes technical information about the implementation of our technique and some performance measurements. In Section 12 we show the obtained results and compare them to other methods of performing occlusions. Finally, in Section 13 we discuss some limitations of the proposed approach as well as provide some pointers to what could be improved in the future.

2 PROBLEM FORMULATION

We will tackle the occlusion handling problem as an alpha matting problem. Alpha matting addresses the problem of extracting foreground objects from static images or video sequences. Precisely separating a foreground object from the background can be achieved by estimating the alpha value for each pixel individually. An image I can be mathematically described as a combination of the foreground image F and the background image B:

$$I = \alpha F + (1 - \alpha)B \tag{1}$$

where α defines the alpha matte (with $\alpha \in [0, 1]$). We can use alpha matting to determine the alpha matte of the scene. We have to precisely separate the foreground objects (parts of the real scene that occlude virtual objects) from the background (objects that lie in the back of virtual objects). By calculating the alpha matte we can then draw the scene using Equation (1) to combine virtual and real objects into a single image. Most alpha matting problems require a user-specified trimap to use as a starting point. A trimap is an image that segments the scene into three non-overlapping regions: definitely background, definitely foreground and unknown regions where it is not possible to determine if the real objects are behind or in front of the virtual objects. It can be concluded that this problem has two main challenges: automatic generation of a trimap based on the relation between real and virtual objects and estimation of the alpha matte. Both the generated trimap should be precise and the unknown regions should be as small to ensure good results.

3 RELATED WORK

The main problems associated with this subject are how to improve the rough depth data captured by the sensor and study different methods for performing occlusion in the context of AR. We also researched about the alpha matting problem used to separate the background and foreground of images in different contexts. Finally we also looked at ways of generating trimaps without user input and in real time.

3.1 Occlusion Handling

Several approaches have been proposed to solve the occlusion problem. Here we present our technique and discuss the results obtained by each approach. According to Hebborn et al. [Hebborn et al. 2017] there are three main approaches used to solve the occlusion problem: model-based, object-based and depth based.

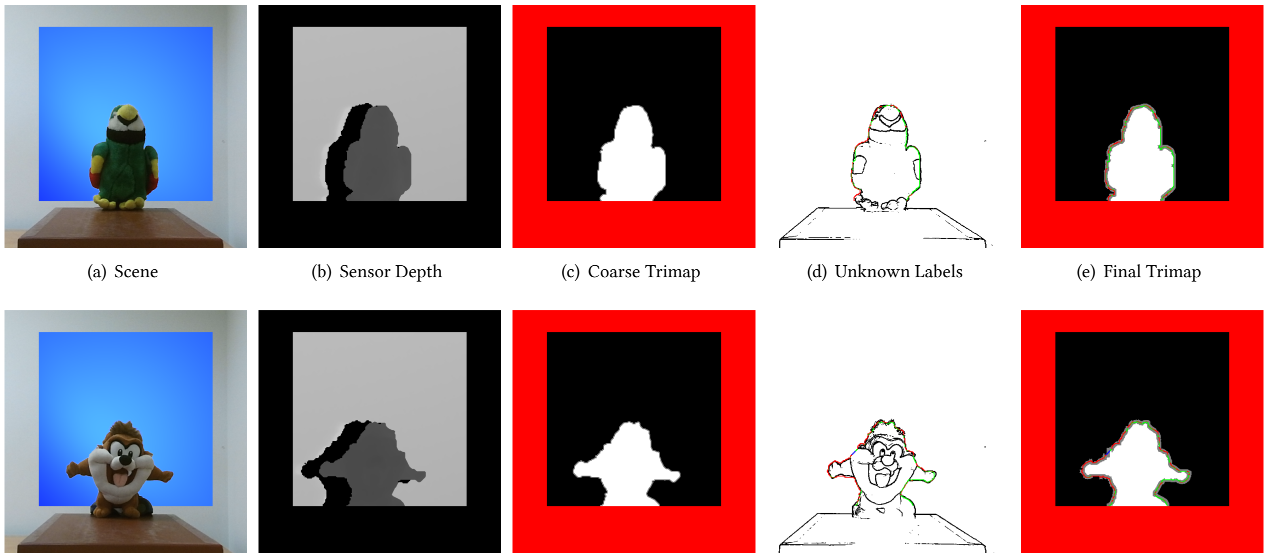
3.1.1 Model-based Methods. These rely on having a 3D model of the scene or having a model of the occluding object. With the model a simple depth test can be performed by aligning real and virtual objects via tracking. The 3D model can be obtained directly from existing resources, built with modeling software or reconstructed using cameras and sensors. Reconstruction algorithms [A. Newcombe et al. 2011; Izadi et al. 2011; Whelan et al. 2015] can reconstruct a semi dense and accurate model but the boundaries are still often noisy. According to Shah *et al.* [Shah et al. 2012] model-based methods are appropriate when the complexity of the scene is low and when it is easy to obtain 3D models of the objects involved.

Overall it can be concluded that this method only works in simple scenes and the quality of the results when using these methods is heavily influenced by the quality of the models or reconstructions as well as the quality of the tracking. Furthermore, this category of methods is usually not able to deal with with dynamic occlusions or deformable objects.

3.1.2 Object-based Methods. These commonly use a contour of the occluding object to handle occlusions. In Lepetit's *et al.* work [Lepetit and marie odile 2000] the user specifies the occluding objects in key-views and then occlusions in intermediate views can be generated automatically. Tian *et al.* [Yuan et al. 2010] proposes a real-time approach which is divided into three steps: selection, tracking and occlusion handling. First the user specifies the occluding object by using an interactive segmentation method. The contour of the object is then tracked in real-time in subsequent frames. In the occlusion handling step the virtual object is drawn on top of the real world image and then all the pixels of the tracked object are redrawn on top of the virtual object.

The main problem of this approach is that it only works if the relative position between real and virtual objects does not change, i.e., if we assume that one virtual object is in front of a real object we cannot move the real object behind that virtual object or the illusion will be broken.

3.1.3 Depth-based Methods. These can handle occlusions for unknown objects without any user input. Nothing about shape, size or position needs to be known previously for depth-based methods to work. They rely on external sensors to capture depth information of the real world such as depth sensors [Du et al. 2016; Leal et al. 2013] or stereo cameras [Schmidt et al. 2002] and then use the depth information to compute occlusion. Although this approach might seem more versatile and powerful there are some drawbacks: depth sensors still produce noisy depth maps and sometimes are not able to capture relevant information. Sensors do not deal well with shadows, are limited by their effective range and cannot perceive reflective or transparent materials. Additionally, camera depth maps are low resolution and not aligned with the its color image.


(a) Scene (b) Sensor Depth (c) Coarse Trimap (d) Unknown Labels (e) Final Trimap

(f) Scene (g) Sensor Depth (h) Coarse Trimap (i) Unknown Labels (j) Final Trimap

Figure 1: Adaptive Trimap Generation. d) Unknown labels on top of color boundaries, e) Unknown labels on top of final trimap

The main problem of this approach is dealing with the problems associated with depth sensors but it is possible to overcome these issues by preprocessing the incoming depth data using different techniques, which we discuss in the next section.

3.2 Improving and Enhancing Depth Maps

To solve the issues of the depth sensors researchers have come up with ways to improve the quality of the depth data by using software. For example, Leal-Meléndrez *et al.* [Leal et al. 2013] uses an inpainting algorithm to fill holes in the depth maps by expanding known regions into unknown regions. Schmidt *et al.* [Schmidt et al. 2002], proposed a method to obtain good quality dense disparity maps (depth map) from stereo images with focus on sharp edges. Du *et al.* [Du et al. 2016], improves depth maps with sharp edges aligned to the edges of the color map. They extract edges from the depth map and from the color map and then use an edge snapping technique to improve the consistency between both. This technique produces good results when the occluding object has well defined color edges which makes it easier to separate the object from the background but has trouble when this condition is not present. Another downside is that this method is very time consuming in higher resolutions.

Since depth maps are essential to many applications in computer vision researchers have come up with various algorithms to improve the quality of the maps. Edge-aware filters are useful for occlusion handling because they smooth images but preserve the edges. Some examples include bilateral filter [Tomasi and Manduchi 1998], joint bilateral filter [Kopf et al. 2007] and guided filter [He et al. 2013]. Schmeing and Jiang [Schmeing and Jiang 2014] proposed a method to aligns depth- with color- edges based on color segmentation.

3.3 Alpha Matting and Trimap Generation

Matting is a known and well-studied technique that is used in many image and video editing applications. For occlusion handling the most relevant methods are the ones where the background is arbitrary and unknown. Alpha matting can be computed using color information, depth information or both. Color-based methods that are relevant for occlusion handling can be divided into two main types: sampling-based and propagation-based.

3.3.1 Sampling-based methods. These [Berman et al. 2000a,b; Ruzon and Tomasi 2000] assume that the true foreground and background colors of an unknown pixel can be estimated from known foreground and background samples. Unknown alpha values are then calculated by solving the inverse composition equation. According to Wang and Cohen [Wang and Cohen 2007a] the results are dependent on the quality of the trimap and selected samples. Sampling-based methods produce good results when the foreground and background have distinct color distributions which simplifies the process of selecting sample colors. Propagation-based methods [Levin et al. 2008; Sun et al. 2004] assume that foreground and background colors are locally smooth and solve the problem by propagating know alpha values into unknown regions. According to Wang and Cohen [Wang and Cohen 2007b] these methods produce good results for simple background and foreground patterns, but the quality quickly degrades in more complex environments.

Several techniques combine the two approaches to achieve high quality results. These techniques collect pairs of foreground and background samples and estimate the alpha values in a global optimization problem, by selecting the best pairs available. This approach can be divided into two problems: 1) selection of suitable samples and 2) definition of a good cost function to use in the optimization process. Wang and Cohen [Wang and Cohen 2007b] generate the samples from the nearest boundary pixels and Johnson *et al.* [Johnson et al. 2016] generate the samples from the closest super pixels. After the selection of sample pairs an objective function is used to calculate the best pair and generate the corresponding alpha value [S. L. Gastal and Oliveira 2010; Wang and Cohen 2007b].

3.3.2 Depth-based Methods. Depth-based methods use depth information to solve the problem of generating high quality trimaps [Lu and Li 2012; Wang et al. 2007]. These start by dividing the foreground and background into a binary map. Then they discover unknown regions by applying erosion and dilatation to the foreground. The separation between foreground and background can be found via a user-defined plane [Wang et al. 2007] or by a segmentation algorithm like k-means [Zhu et al. 2009]. Cho *et al.* [Cho et al. 2011] proposed an adaptive dilation according to the fuzziness of the foreground object. This approach allows fuzzy objects to be covered by larger unknown regions than objects with sharp edges.

3.3.3 Color and depth-based Methods. These combine the information of color and depth to achieve the best quality. Hebborn *et al.* [Hebborn et al. 2017] generate an adaptive trimap automatically using depth and color information. They start by generating a coarse trimap using depth information. Then the pixels on the unknown regions are labeled according to their position relative to the edges of the color image. Finally using the information computed previously an adaptive dilation is applied to extend unknown regions in a way that that both depth and color boundaries are covered while keeping the unknown regions as small as possible. After the refined trimap is found, known foreground and background regions of the color image are propagated toward the unknown regions. Finally, the alpha matte is estimated through a sample-based method where local samples are collected from the foreground and background and the best samples are selected using an objective function.

4 OUR APPROACH

Our technique takes advantage of color and depth information provided by a sensor such as the Kinect. The color and depth information are then processed in several steps to achieve the final alpha matte and combine the virtual objects with the real world scene. The main steps in this process are as follows:

Virtual Scene Rendering. Renders the virtual scene into textures. The color is rendered into a texture and the depth information is rendered to a second texture.

Depth Data Smoothing. Filters the depth information provided by the sensor to remove noise and improve the quality of the depth map.

Coarse Trimap Generation. Uses the depth information provided by the sensor and the depth information of the virtual scene to generate an initial trimap.

Adaptive Trimap Dilation. Uses color information captured by the sensor to further refine the initial trimap. During this stage unknown areas are dilated to cover possible fuzzy areas or areas where the depth information is inaccurate or invalid.

Foreground and Background Color Propagation Propagates the colors of the known foreground and background regions into the unknowns areas of the trimap.

Alpha Estimation. Uses the colors of the expanded foreground and background images to find a good alpha value for each pixel of the unknown areas.

Alpha Matte Smoothing. Filters the generated alpha matte to reduce visual noise and small imperfections.

Compositing. Combines the color images of the virtual scene and real world using the alpha matte previously generated.

4.1 Depth Data Smoothing

Due to the fact that the depth data captured by sensors is typically noisy we first try to remove as much noise as possible using a 5 by 5 median filter. This filter improves the quality of the depth map by smoothing the edges of the objects in the scene.

4.2 Coarse Trimap Generation

In this step we split the scene into foreground, background and unknown regions. The trimap represents the three possible types of regions by coloring each region with a diferent color: the foreground is painted with white, the background with black and the unknown regions with grey. A fourth color, red, is used to paint areas which are not covered by the virtual objects and, as such, there is no need to find whether the virtual objects are occluded.

First we generate an initial trimap where the known foreground and background regions are found by applying a simple depth test between the scene depth data and the virtual depth data. If the sensor's depth data is invalid, which occurs due to shadowing effects, the region is marked as background.

Finally, to find the unknown regions, which are neither in the background or in the foreground, we apply a 3 by 3 sobel filter to the initial trimap. This filter detects the transitions between the foreground and the background regions which are marked as unknown. The sobel filter also provides us with information regarding the direction of the transitions which will be important in the next steps.

4.3 Adaptive Trimap Dilation

If we only took the depth information into account then we would obtain wrong alpha estimations because sometimes we would be taking samples from the foreground as if they were samples from the background and vice versa.

To solve this issue We need to expand the unknown regions in such a way that they cover not only the transitions in the depth map but also transitions in the color image. Furthermore, we want to keep these regions as small as possible otherwise they could be expanded until enough information was lost and make alpha estimation impossible.

To overcome these problems we take the boundaries of the depth map as an initial estimation and then expand the unknown regions towards the color boundaries. To achieve this goal we must first find the relative positions between the depth map boundaries and the color map boundaries so we can expand the unknown regions in the right direction.

4.4 Labeling Unknown Regions

This step labels all pixels in the unknown regions according to their position in relation to the boundaries of the color map.

After we have found the boundaries of the color image we can categorize all pixels of the unknown regions into three categories: *front-half space*, *back-half space* and *no edge*. An unknown pixel i is labeled with *front-half space* if i lies in the front-half space of an edge in the color image. Conversely, if the unknown pixel i lies in the back-half space of an edge in the color image it is labeled as *back-half space*. If there are no edges in the color image around the unknown pixel i the pixel i is labeled as *no edge*.

To determine whether an unknown pixel is in the back-space or front-space of an edge or if it has no nearby edges in the color image we must look for pixels that belong to the boundaries of the color image in a window around the unknown pixel. If the number of color edge pixels is bellow a certain threshold then the unknown pixel is marked as *no edge*. If there are enough color edge pixels around the unknown pixel then we can find if the unknown pixel is in front or behind the color edge pixels by computing the scalar product between the direction of the unknown pixel gradient and the direction from the unknown pixel to the color edge pixel as explained in Figure 2. This procedure and its results are explained in detail in [Silva 2018].

4.5 Adaptive Dilation

In order to expand the unknown regions we took inspiration from Chen *et al.* [Chen et al. 2012]. We expand the unknown regions starting in the pixels marked previously as unknown and propagating them until we reach the boundaries of the color image. We use the information that was computed in the previous step in order to find the direction of expansion.

Furthermore when an initial unknown pixel is marked as *no edge* we expand the unknown region in all directions around that pixel. The amount of dilation applied in this case depends on the number of no edge pixels found in that region of the trimap. This mechanism allows us to cover areas where the information of the color is fuzzy, for example when fur or hair are present.

4.6 Foreground and Background Propagation

To get a good alpha estimation we need to find good samples of the known foreground and background colors. We chose to propagate the known values into the unknown areas. This step allows us to later use the propagated colors to realize a universal search in a window around each unknown pixel.

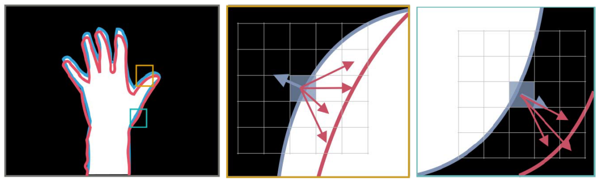


Figure 2: Labeling Process. (a) unknown regions (blue) overlaid with boundaries in the color image (red), (b) the unknown pixel lies in the front-half space of the color boundaries, (c) the unknown pixel lies in the back-half space of the color boundary.

We took inspiration from Hebborn *et al.* [Hebborn et al. 2017] to solve this problem. The idea of the technique is to successively blur the known values into the unknown regions and write back only the blurred values where the image was empty before. By repeating this process several times we manage to diffuse the color of the known boundaries into the unknown regions. This technique is similar to the known hole filling method proposed by Davis *et al.* [Davis et al. 2002]. To increase the speed of the process the blurring occurs in lower resolution images as seen in pyramid-based filtering algorithms [J Burt 1981; Ogden et al. 1985]. These approaches build an image pyramid by downscaling the original image down to a specified level. The required operations are then performed in the lowest resolution image, which is much smaller than the original image. And finally the processed low resolution image is upscaled back to the original size.

The process is applied to both the foreground and background known values. So, to simplify, we will only explain how it works for the foreground. Figure 1 illustrates the steps of a single iteration of the image diffusion. The more times we repeat this algorithm the smoother the propagation colors become.

The first step of the diffusion is to copy the foreground image into a new image S (line 1). Then we must initialize all the alpha values of the image S (lines 2-8). We then build the image pyramid and initialize the first level with all the known foreground colors and set their α channel to 1 (line 9). We create the lower resolution levels of the image pyramid by sequentially down-sampling the previous level using a simple linear filter. In the end we clear all the levels except the lowest one. Next we start from the lowest level of the pyramid and start to rebuild the pyramid by up-sampling the lowest level using a quadratic B-spline interpolation. It is also important to point out that the factors of the quadratic interpolation are weighted by the α values so that only known values are taken into account. We repeat this process until we reach the original resolution. This allows for a fast smoothing of the original image and for the known values to propagate past the original boundaries of the starting image. To finish off we must combine the original image with the new blurred one (lines 11-17). The goal is to copy only new color values where previously there were none. To achieve a smooth transition between diffusion steps a constant n is used to control how the new and previous colors are combined.

The propagation extent depends both on the height l of the image pyramid and on diffusion steps i. If the pyramid height increases then the propagated area grows. Thus, the more diffusion steps performed, the smoother the propagated colors become. Figure 3 shows the final result of the propagation algorithm according to pyramid height l and diffusion steps i.

For each pixel created during the image propagation algorithm we save the iteration i where that pixel was created. We later use this information to measure how far that color is from the original pixels in the alpha estimation process.

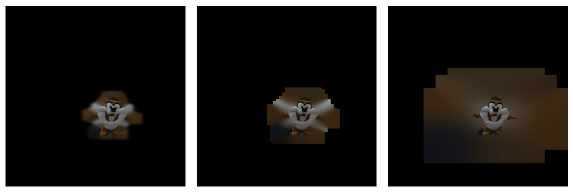
4.7 Alpha Estimation

To estimate the alpha matte we take pairs of samples from the expanded foreground and background images and try to find the pair of values that best represent the color of the image when linearly combined. The best pair is defined by an objective function

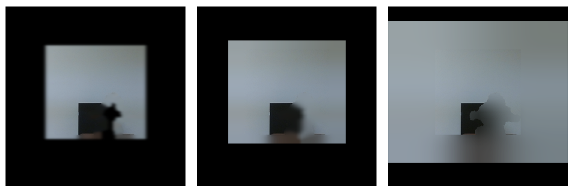
Algorithm 1: Single step of the diffusion algorithm

```
copy input image $F$ with 4 channels to image $S$;
for each pixel $p^s$ of the image $S$ do
    if $p^s_a > 0$ then
        $p^s_a \leftarrow 1$;
    else
        $p^s_a \leftarrow 0$;
    end
    end if;
end
end for;
build image pyramid of $S$ with $l$ levels;
go top-down the pyramid to smooth $S$;
for each pixel $p^f$ of the image $F$ do
    if $p^s_a > 0$ then
        $w_{i-1} \leftarrow \frac{p^f_a}{n}$ ► calculate normalized weight;
        $p^f_{rgb} \leftarrow w_{i-1} p^f_{rgb} + (1 - w_{i-1}) p^s_{rgb}$;
        $p^f_a \leftarrow min(p^f_a + 1, n)$ ► increase and clamp alpha;
    end
    end if;
end
end for;
```

that considers both a color cost and a propagation cost. The color cost is the difference between the linear combination of the two colors with a certain α value and the real color captured by the camera. The propagation cost takes into account the iteration i when the colors were generated in the expanded images. We can estimate the α value of a given sample pair (F_k, B_l) via Eq. 2, where Ip is the color of the unknown pixel in the image.



(a) Foreground: $l = 2$ and $i = 20$ (b) Foreground: $l = 4$ and $i = 5$ (c) Foreground: $l = 6$ and $i = 4$



(d) Background: $l = 2$ and $i = 20$ (e) Background: $l = 4$ and $i = 5$ (f) Background: $l = 6$ and $i = 4$

Figure 3: Propagation of Foreground and Background. Results for different pyramid levels l and diffusion steps i

$$\alpha_p(F_k, B_l, I_p) = \frac{(I_p - B_l)(F_k - B_l)}{||F_k - B_l||^2} \tag{2}$$

4.7.1 Objective Function. When we want to estimate the alpha value of a certain unknown pixel $p_{i,j}$ we try to minimize the objective function for all the possible pairs of foreground and background colors in a window of $n \times n$ pixels around p ($\{F_{i-n,j-n}, F_{i-n+1,j-n+1}, ..., F_{i+n,j+n}\}$, $\{B_{i-n,j-n}, B_{i-n+1,j-n+1}, ..., B_{i+n,j+n}\}$). For each of those pairs (F_k, B_l) we solve the objective function:

$$(F_k, B_l) = wC_{col}(F_k, B_l, I_p) + C_{pro}(F_k, B_l) \tag{3}$$

where w is the weight of the color cost, $C_{col}(F_k, B_l, I_p)$ is the color cost (Eq. 4) and $C_{pro}(F_k, B_l)$ is the propagation cost (Eq. 5).

After we find the best pair we can estimate the α value using Eq. 2. The alpha value of the known values is set to 1 where the foreground is known and to 0 where the background is known.

4.7.2 Color Cost Function. Measures how well a sample pair is able to represent value I_p of the image by a linear combination of those pixels. Thus, if a pair of foreground and background colors can approximate the intensity I_p, the associated cost should be small:

$$C_{col}(F_k, B_l, I_p) = ||I_p - (\alpha F_k + (1 - \alpha) B_l)|| \tag{4}$$

where α is the estimated value for the pair F_k, B_l using Eq. 2.

4.7.3 Propagation Cost Function. To reduce the impact of wrongly propagated colors we record when in the process of the image propagation a specific pixel color was created. That way, colors created during the first steps should have a low propagation cost, as defined by Eq. 5, since their values are closer to known RGB values in the image.

$$C_{pro}(F_k, B_l) = \frac{d(F_k) + d(B_l)}{2d_m} \tag{5}$$

In Eq. 5 $d(p)$ returns the number of the iteration $d_i \in [0, d_{m-1}]$, in which the color p was created during image propagation, where d_m is the total iterations of the image propagation algorithm.

4.8 Alpha Matte Smoothing

After we compute alpha values for all pixels we smooth them by applying a 5×5 low pass filter to the final alpha matte, to both remove unwanted noise and improve its quality.

5 IMPLEMENTATION AND PERFORMANCE

To implement our approach we opted to use C++, OpenGL 4.3 and the OpenGL Toolkit (GLUT). All the computational steps take place in the GPU and are programmed in OpenGL Shader Language. The search for good sample pairs in the alpha estimation step is also done in the GPU to be as quick as possible. We perform each computational step as a rendering pass mapping the result into a texture using frame buffers. That texture is used in the next computational steps when needed. We used the *Kinect V2* sensor to capture color and depth information. The performance measurements were obtained for a resolution of 1920 by 1080 in both color and depth data. The propagation algorithm ran in a window of 1024 by 1024 around the area of interest (the center of the virtual object).

Table 1: Performance measurements of the main steps of our method. With 4 pyramid levels *l* and 6 diffusion steps *i*.

Number of Unknown Pixels	6617	6978	13973
Trimap Generation (ms)	0.1228	0.1286	0.1333
Propagation (ms)	1.328	1.406	1.368
Alpha Estimation (ms)	0.02433	0.02575	0.03592
Total (ms)	1.576	1.541	1.518

5.1 Performance Evaluation

The computer used to perform the measurements was equipped with an equipped with an Intel i7-6700 3.40GHz processor and an NVIDIA Graphics Card. The measurements obtained are the average value obtained after executing 1000 cycles of the algorithm. As seen in Table 1 the number of unknown pixels do not cause significant changes in the overall performance of the algorithm. As expected, the amount of unknown pixels affect mainly the performance of the alpha estimation step, since this step depends directly on the amount of unknown pixels. Anyways, it can be seen that the most computational intensive step is the propagation of the known color values of the foreground and background into the unknown regions. The propagation step is responsible for almost 90% of the total time. This step, shown in Table 2, increases linearly with pyramid height l and the diffusion steps i. The algorithm needs around 1.5 ms for generating each frame which easily beats the real-time performance benchmark of 30 frames per second. Even with all the overhead of loading the sensors data into the GPU and the overhead associated with OpenGL our technique is able to process 40 to 50 frames per second.

6 RESULTS

To show our approach working we chose three representative scenarios: one scenario where the virtual object is simple (a plane) and the background and foreground colors are substantially different, a second scenario where we keep the same simple virtual object but where the foreground and background have similar colors and a third scenario where we use a more complex virtual object.

To compare results we implemented an occlusion computing prototype that uses only the depth to compute occlusions with a simple depth test. We then took the simple raw depth method and improved it by applying a median filter to the depth map and by smoothing the resulting alpha matte with a low pass filter. Finally, we also compare our results with the results from the original alpha matting algorithm [Hebborn et al. 2017]. The comparison results can be seen in Figure 4.

7 LIMITATIONS

Our method does not perform very well when the foreground and background have similar colors. When this scenario occurs we are not able to extract information about the color boundaries of the image and so we must rely only on depth data which is, most of the times, not accurate enough.

Another limitation is dealing with very narrow regions between objects of the scene. Sometimes it is not possible to accurately distinguish between foreground and background in these regions.

Table 2: Performance measurements for different pyramid levels *l* and diffusion steps *i*.

Diffusion Steps i	Pyramid Levels l	Time in ms
2	10	0.8658
2	20	1.093
4	5	0.7029
6	4	0.7678
10	2	0.965
20	2	1.205

This happens because the depth data has not enough resolution to detect changes in depth values in narrow regions. As the depth map edge filter fails to detect boundaries in these regions they are not not marked as unknown and are treated as foreground.

Another problem which sometimes affects the final result is the flickering that happens in the depth data captured by the sensor we used. We tried to minimize this problem by smoothing the alpha matte both spatially and temporally. We were able to improve the final result by applying a low pass filter to the alpha matte but we were not able to implement any kind of temporal filtering without causing a significant performance drop.

8 CONCLUSIONS

In this work we improved the algorithm proposed by Hebborn et al. [Hebborn et al. 2017] by using different filters to treat depth data captured by the sensor, by tweaking the adaptive dilation step and by adding a final post processing step to smooth and improve the produced alpha matte. These changes make the final result both more stable and aesthetically pleasing.

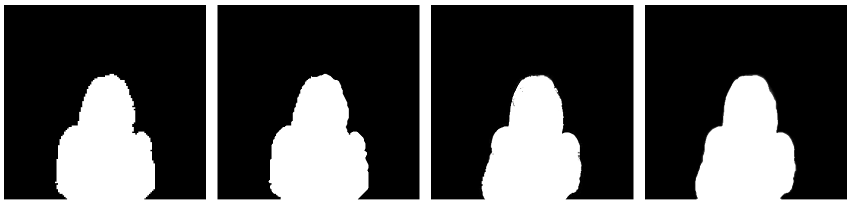
9 FUTURE WORK

One possible improvement to the current method is to apply more advanced optimization techniques in the alpha estimation step. Currently we use a primitive local search to find the best pairs. Maybe one could apply artificial intelligence techniques in this step to come up with a better approach to estimate the alpha matte. These techniques could even help improve object selection in other VR/AR contexts [Mendes et al. 2017].

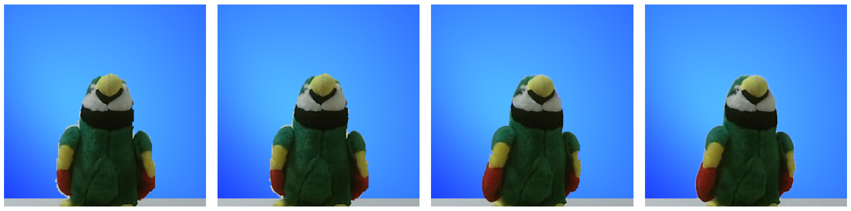
Another possible improvement is to apply temporal stabilization to the final result using information from the previous frames. We tried to apply an averaging operation between the current and the last frames but were not able to get any improvements without slowing down the final result. Maybe as the graphic cards get faster this technique could be viable. If not, maybe machine learning approaches could also be used to reduce the flickering in the depth information that sometimes bleeds into the final result. Anyways, we feel like most of the issues affecting this approach could be improved by simply using more accurate sensors which will likely be available in the near future, or by learning better filters through data-driven techniques.

ACKNOWLEDGMENTS

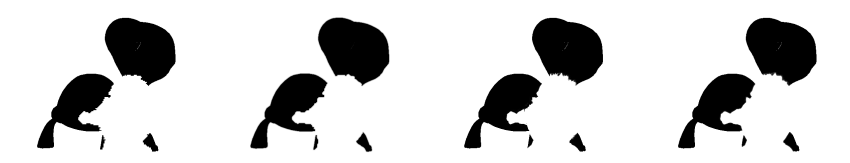
This project was funded by FCT through grants IT-MEDEX PTDC /EEISII/6038/2014 and UID/CEC/50021/2019.



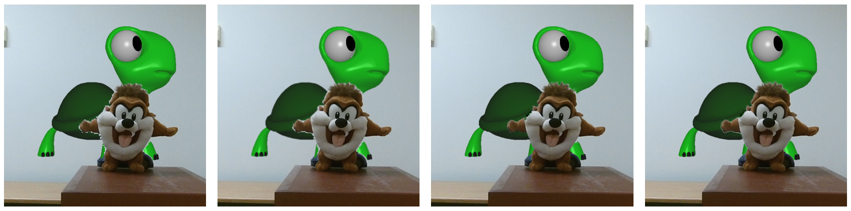
(a) Raw depth (b) Improved (c) Original (d) Ours



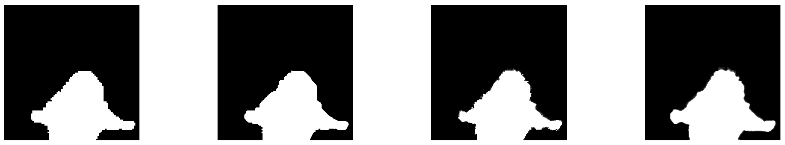
(e) Result of (a) (f) Result of (b) (g) Result of (c) (h) Result of (d)



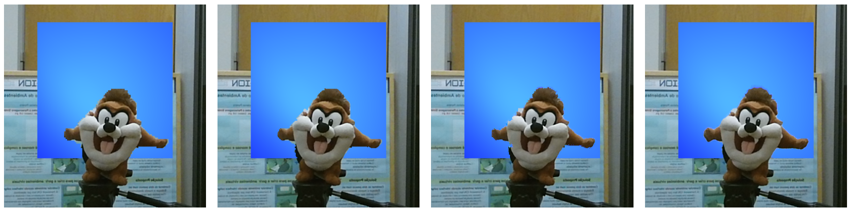
(i) Raw (j) Improved (k) Original (l) Ours



(m) Result of (i) (n) Result of (j) (o) Result of (k) (p) Result of (l)



(q) Raw (r) Improved (s) Original (t) Ours



(u) Result of (q) (v) Result of (r) (w) Result of (s) (x) Result of (t)

Figure 4: Results of applying the four algorithms. Left column (a,l,q) shows the simple raw depth method, that we improved by applying a median filter and smoothing the alpha matte via a low pass filter to reduce jagged edges (b,j,r) and yield a more pleasing result (f,n,v). On the right, we compare our technique with [Hebborn et al. 2017] (cols c, q, s). Although not visible in the still pictures, our method (d, l, t) produces more stable results (h, p, x). This is better illustrated in the supplemental video provided.

REFERENCES

P A. Fortin and P Hebert. 2006. Handling Occlusions in Real-time Augmented Reality : Dealing with Movable Real and Virtual Objects. In *Third Canadian Conference on Computer and Robot Vision (CRV)*. 54.

Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. 2011. DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision*. 2320–2327.

A Berman, A Dadourian, and P Vlahos. 2000a. Method for removing from an image the background surrounding a selected object. (January 2000).

A Berman, P Vlahos, and A Dadourian. 2000b. Comprehensive method for removing from an image the background surrounding a selected object. (October 2000).

David Breen, Eric Rose, and Ross T. Whitaker. 1995. *Interactive Occlusion and Collision of Real and Virtual Objects in Augmented Reality.* Technical Report. European Computer-Industry Research Centre.

L. Chen, H. Lin, and S. Li. 2012. Depth image enhancement for Kinect using region growing and bilateral filter. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. 3070–3073.

J. Cho, T. Yamasaki, K. Aizawa, and K. H. Lee. 2011. Depth video camera based temporal alpha matting for natural 3D scene generation. In *2011 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*. 1–4.

J. Davis, S. R. Marschner, M. Garr, and M. Levoy. 2002. Filling holes in complex surfaces using volumetric diffusion. In *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*. 428–441. https://doi.org/10.1109/TDPVT.2002.1024098

C. Du, Y. Chen, M. Ye, and L. Ren. 2016. Edge Snapping-Based Depth Enhancement for Dynamic Occlusion Handling in Augmented Reality. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 54–62.

Jan Fischer, Holger Regenbrecht, and Gregory Baratoff. 2003. Detecting dynamic occlusion in front of static backgrounds for AR scenes. In *EGVE '03 Workshop on Virtual environments*. 153–161.

Kenichi Hayashi, Hirokazu Kato, and Shogo Nishida. 2005. Occlusion detection of real objects using contour based stereo matching. 180–186.

K. He, J. Sun, and X. Tang. 2013. Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 6 (June 2013), 1397–1409.

A. K. Hebborn, N. Höhner, and S. Müller. 2017. Occlusion Matting: Realistic Occlusion Handling for Augmented Reality Applications. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 62–71. https://doi.org/10.1109/ISMAR.2017.23

Bradley Herbert, Barrett Ens, Amali Weerasinghe, Mark Billinghurst, and Grant Wigley. 2018. Design considerations for combining augmented reality with intelligent tutors. *Computers & Graphics* 77 (2018), 166 – 182. https://doi.org/10.1016/j.cag.2018.09.017

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard A. Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew J. Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST)*. 559–568.

Peter J Burt. 1981. Fast Filter Transforms for Image Processing. *Computer Graphics and Image Processing* 16 (May 1981), 20–51.

Jubin Johnson, Ehsan Shahrian, Hisham Cholakkal, and Deepu Rajan. 2016. Sparse Coding for Alpha Matting. In *IEEE Transactions on Image Processing*, Vol. 25. 1.

Johannes Kopf, Michael Cohen, Dani Lischinski, and Matt Uyttendaele. 2007. Joint bilateral upsampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26 (July 2007), 96.

Adrian Leal, Leopoldo Altamirano Robles, and Jesus Gonzalez. 2013. Occlusion Handling in Video-Based Augmented Reality Using the Kinect Sensor for Indoor Registration. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. 447–454.

Vincent Lepetit and Berger Marie-Odile. 2000. A Semi-Automatic Method for Resolving Occlusion in Augmented Reality. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2. 2225–2230.

Vincent Lepetit and Berger marie odile. 2000. A Semi-Automatic Method for Resolving Occlusion in Augmented Reality. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2225–2230.

Anat Levin, Dani Lischinski, and Yair Weiss. 2008. A Closed-Form Solution to Natural Image Matting. In *IEEE transactions on pattern analysis and machine intelligence*, Vol. 30. 228–42.

Daniel Simões Lopes and Joaquim A. Jorge. 2019. Extending medical interfaces towards virtual reality and augmented reality. *Annals of Medicine* 51, sup1 (2019), 29–29. https://doi.org/10.1080/07853890.2018.1560068 arXiv:https://doi.org/10.1080/07853890.2018.1560068

Ting Lu and Shutao Li. 2012. Image matting with color and depth information. In *International Conference on Pattern Recognition*. 3787–3790.

Matthias M. Wloka and Brian G. Anderson. 1995. Resolving Occlusion in Augmented Reality. In *Symposium on Interactive 3D graphics (I3D)*. 5–12.

Daniel Mendes, Daniel Medeiros, Maurício Sousa, Eduardo Cordeiro, Alfredo Ferreira, and Joaquim A. Jorge. 2017. Design and evaluation of a novel out-of-reach selection technique for VR using iterative refinement. *Computers & Graphics* 67 (2017), 95 – 102. https://doi.org/10.1016/j.cag.2017.06.003

Joan Ogden, Edward Adelson, James Bergen, and Peter J. Burt. 1985. Pyramid-based Computer Graphics. *RCA engineer* 30 (September 1985), 4–15.

Bernhard Preim and Patrick Saalfeld. 2018. A survey of virtual human anatomy education systems. *Computers & Graphics* 71 (2018), 132 – 153. https://doi.org/10.1016/j.cag.2018.01.005

M. A. Ruzon and C. Tomasi. 2000. Alpha estimation in natural images. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000*, Vol. 1. 18–25.

Eduardo S. L. Gastal and Manuel Oliveira. 2010. Shared Sampling for Real-Time Alpha Matting. In *Eurographics 2010*, Vol. 29. 575–584.

Michael Schmeing and Xiaoyi Jiang. 2014. Edge-aware depth image filtering using color segmentation. *Pattern Recognition Letters* 50 (December 2014), 63–71.

J. Schmidt, H. Niemann, and S. Vogt. 2002. Dense disparity maps in real-time with an application to augmented reality. In *Sixth IEEE Workshop on Applications of Computer Vision, 2002. (WACV 2002). Proceedings*. 225–230.

Susanne Schmidt, Gerd Bruder, and Frank Steinicke. 2019. Effects of virtual agent and object representation on experiencing exhibited artifacts. *Computers & Graphics* 83 (2019), 1 – 10. https://doi.org/10.1016/j.cag.2019.06.002

Paul D. Schmirler, Thong T. Nguyen, Alex L. Nicoll, and David Vasko. 2018. Virtual reality and augmented reality for industrial automation. PATENT US20180131907A1. https://patents.google.com/patent/US20180131907A1/en

M. M. Shah, H. Arshad, and R. Sulaiman. 2012. Occlusion in augmented reality. In *2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012)*, Vol. 2. 372–378.

Ricardo Silva. 2018. *Dynamic Occlusion Handling for Real-Time AR Applications.* Master's thesis. Instituto Superior Técnico da Universidade de Lisboa, Avenida Rovisco Pais s/n 1049-001, Lisboa, Portugal.

Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. 2004. Poisson matting. In *ACM SIGGRAPH*, Vol. 23. 315–321.

C. Tomasi and R. Manduchi. 1998. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. 839–846.

Jue Wang and Michael Cohen. 2007a. Image and Video Matting: A Survey. *Foundations and Trends in Computer Graphics and Vision* 3 (January 2007), 97–175.

J. Wang and M. F. Cohen. 2007b. Optimized Color Sampling for Robust Matting. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.

O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang. 2007. Automatic Natural Video Matting with Depth. In *15th Pacific Conference on Computer Graphics and Applications*. 469–472.

Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. 2015. ElasticFusion: Dense SLAM Without A Pose Graph. (July 2015).

Tian Yuan, Guan Tao, and Wang Cheng. 2010. Real-Time Occlusion Handling in Augmented Reality Based on an Object Tracking Approach. In *Sensors*, Vol. 10.

J. Zhu, Miao Liao, R. Yang, and Zhigeng Pan. 2009. Joint depth and alpha matte optimization via fusion of stereo and time-of-flight sensor. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 453–460.

Jiejie Zhu, Zhigeng Pan, Chao Sun, and Wenzhi Chen. 2010. Handling occlusions in video-based augmented reality using depth information. *Journal of Visualization and Computer Animation* 21 (September 2010), 509–521.

Ezequiel R. Zorzal, Maurício Sousa, Daniel Mendes, Rafael Kuffner dos Anjos, Daniel Medeiros, Soraia Figueiredo Paulo, Pedro Rodrigues, José João Mendes, Vincent Delmas, Jean-Francois Uhl, José Mogorrón, Joaquim Armando Jorge, and Daniel Simões Lopes. 2019. Anatomy studio: A tool for virtual dissection through augmented 3D reconstruction. *Computers & Graphics* (2019). https://doi.org/10.1016/j.cag.2019.09.006